Wireless Chip-Scale Communications for Neural Network Accelerators

A Degree Thesis
Submitted to the Faculty of the
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

by

Robert Guirado

In partial fulfillment
of the requirements for the degree in
Telecommunications Technologies and Services Engineering

Advisor: Sergi Abadal, Tushar Krishna

Atlanta, June 2019

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

Georgia Tech School of Electrical and Computer Engineering
College of Engineering

Abstract

The last decade has witnessed an explosive growth in the use of Deep Neural Networks in fields such as computer vision, natural language processing, medicine or economics. Their achievements in accuracy across so many relevant and different applications exhibit the enormous potential of this disruptive technology. However, this unprecedented performance is closely tied with the fact that their new designs contain much deeper and bigger layer sets, forcing them to manage millions - and in some cases even billions - of parameters. This comes at a high computational and communication cost at the processor level, which has prompted the development of new hardware aimed at handling such large computing expense more efficiently, the so called Deep Neural Network accelerators. This work explores the potential of enhancing the performance of these accelerators by introducing Wireless Networks-on-Chip in their design, a novel interconnect paradigm proposed by the research community to overcome some of the communication challenges that manycore systems face. Specifically, both on-chip and off-chip wireless interconnect implementations have been studied and evaluated. In the off-chip case, a theoretical improvement of 13X in the runtime has been achieved, but at the expense of some area and power overheads.

Resum

La darrera dècada ha estat testimoni d'un immens creixement en l'ús de Deep Neural Networks en camps com la visió artificial, processament de llenguatge natural, medicina o economia. Haver aconseguit aquests resultats sense precedents en aplicacions tan rellevants i variades mostra l'enorme potencial d'aquesta tecnologia tan disruptiva. No obstant, aquests èxits van molt lligats al fet de que els nous dissenys contenen moltes més capes i més profundes, cosa que es tradueix en milions - i en alguns casos bilions - de paràmetres. Això suposa un gran cost computacional i de comunicació a nivell de processador, cosa que ha impulsat el desenvolupament de nou hardware que permetin gestionar tal cost de manera més eficient, els anomenats acceleradors de Deep Neural Networks. Aquest projecte explora la potencial millora en rendiment d'aquests acceleradors mitjançant la introducció de Wireless Newtorks-on-Chip al seu disseny, un nou paradigma d'interconnexions proposat per la comunitat científica per a superar alguns dels problemes de comunicació que sistemes manycore han d'afrontar. Específicament, implementacions tant on-chip com off-chip s'han estudiat i evaluat. En el cas off-chip, s'ha aconseguit una millora teòrica de 13X al runtime però amb alguns costos afegits d'àrea i potència.

Resumen

La última década ha sido testigo de un inmenso crecimiento en el uso de Deep Neural Networks en campos como la visión artificial, procesamiento de lenguaje natural, medicina o economía. Haber conseguido estos resultados sin precedentes en aplicaciones tan relevantes y variadas muestra el enorme potencial de esta tecnología tan disruptiva. Sin embargo, estos logros van muy ligados al hecho de que los nuevos diseños contienen muchas más capas y más profundas, lo que se traduce en millones - y en algunos casos billones - de parámetros. Esto supone un gran coste computacional y de comunicación a nivel de procesador, lo que ha impulsado el desarrollo de nuevo hardware que permita gestionar tal coste de manera más eficiente, los llamados aceleradores de Deep Neural Networks. Este proyecto explora la potencial mejora en rendimiento de estos aceleradores mediante la introducción de Wireless Newtorks-on-Chip en su diseño, un nuevo paradigma de interconexiones propuesto por la comunidad científica para superar algunos de los problemas de comunicación que sistemas manycore deben afrontar. Específicamente, implementaciones tanto on-chip como off-chip se han estudiado y evaluado. Se ha conseguido una mejora teórica de 13X en el runtime, pero con algunos costes añadidos de área y potencia.

To my family, friends and girlfriend, who always supported me.

Acknowledgments

First and foremost, I would like to thank professor Tushar Krishna, my research co-advisor in Atlanta, for his tireless passion and every advice he gave me during the project. By providing me a position in his lab, he helped me fulfill my dream of studying in the United States. I also appreciate the chance to work with amazing lab mates such as Hyoukjun, thank you for all your wise words.

I would also like to express my gratitude to my mentor Sergi Abadal, for his patience and guidance, and the dedicated involvement in helping me write this thesis. He encouraged me every time I was stuck and none of this would have been possible without his endless support.

Of course, I must thank both UPC and GeorgiaTech for this mobility opportunity, and especially Eduard Alarcón, who believed in me and gave me a chance to study abroad and live this experience. This thesis also represents the work done during four years to complete my degree. I thank all the professors who taught me and pushed me towards this moment. I must also thank Pol, Cantos, Marc, and many other friends I made throughout the way, for tolerating me and sharing the struggle, to be stronger together.

Getting out of the comfort zone to pursue this adventure required incredible friends to whom one can talk about anything at any time, people who will listen to you and support you when you are feeling down and 8000 km away from home. Thank you, Dani, for being there.
Irene, David, Apolline, Quentin, Rishab, you have been my family in Atlanta. To my lifetime friends back in Barcelona, thank you for caring and putting a smile on me every time I needed it.

Most importantly, none of this could have happened without my family. My parents and my sister, who unconditionally supported me in every life choice I made. My grandparents, who tested their tech skills to skype with me. To all of them, to check that I was alive every two or three days. To Alba, you are the best I know.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Procedure

This thesis has been carried out at the Electrical and Computer Engineering Department of the Georgia Institute of Technology (GeorgiaTech) in Atlanta, specifically in the Synergy Lab [1].
This laboratory, under Professor Tushar Krishna's supervision, focuses on architecting next-generation intelligent computer systems. The research components at Synergy Lab span from system software to microarchitecture, allowing optimizations in every layer.

Therefore, this project is embedded inside the range of projects being carried out at the Synergy lab. The members of the lab include PhD students, Master students and undergraduate students.

Having said that, this project approach - using Wireless Networks-on-Chip - represents a novelty in this lab and thus introduces a new factor to their existing lines of research.
The process will be based on previous designs of other members of the Synergy Lab, by using their available software tools and recommendations, and adding the wireless technology options when appropriate. The main project initial ideas have been provided by Sergi Abadal (UPC), Eduard Alarcón (UPC) and Tushar Krishna (GeorgiaTech).

1.2 Workplan

The following Gantt diagram shows the project development breakdown.

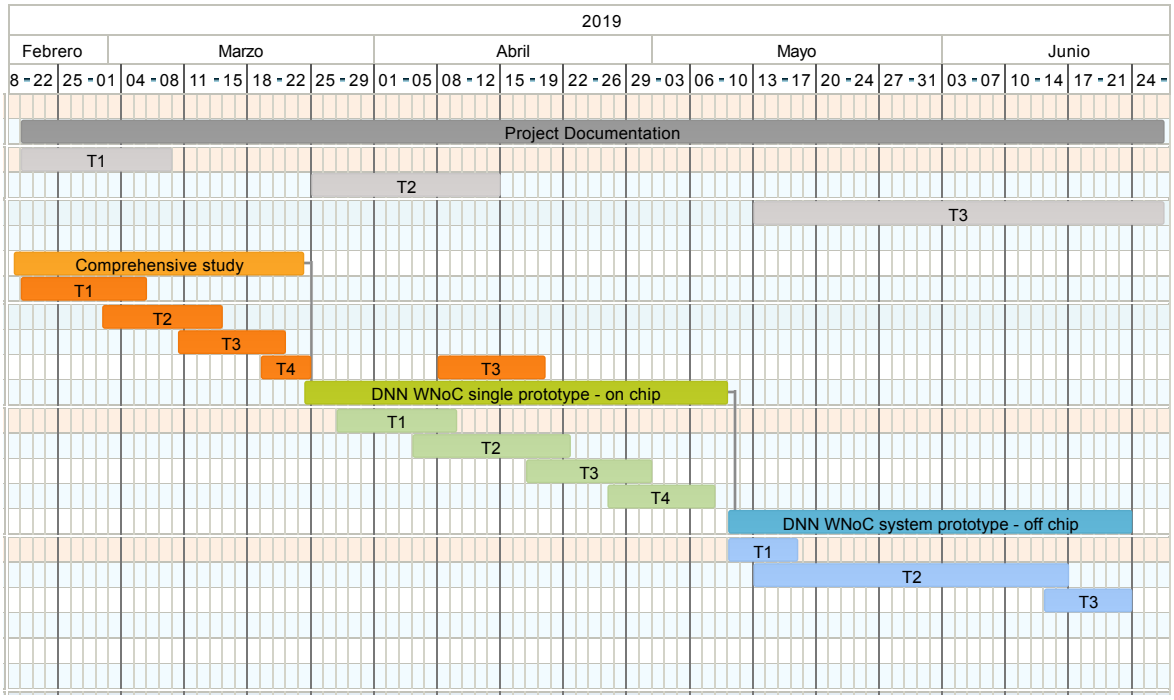


Figure 1.1: *Project Gantt diagram*

Work Package 1: Comprehensive study
Internal task T1: DNN accelerators and dataflow research. Internal task T2: Familiarize with the concepts and software tools. Internal task T3: Full study for each dataflow in every architecture. Internal task T4: Study of WNoC inside a DNN accelerator chip.

Work Package 2: DNN WNoC single prototype
Internal task T1: Transceiver models. Internal task T2: Building blocks creation and simulation. Internal task T3: Iteration of different designs. Internal task T4: Selection of a design based on its performance.

Work Package 3: DNN WNoC system prototype
Internal task T1: Building blocks creation and simulation. Internal task T2: Iteration of different designs. Internal task T3: Specific design.

Work Package 4: Project documentation
Internal task T1: Proposal and Workplan. Internal task T2: Critical design review. Internal task T3: Final report.

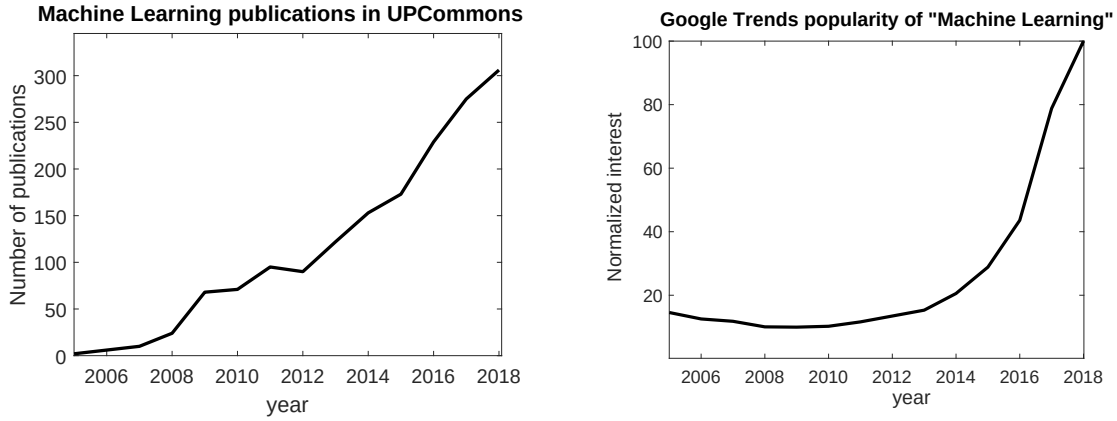
The revision history and approval record can be found in Annex II.

1.3 Framework

Artificial Intelligence (AI) is the area of computer science that studies how to bring intelligence to machines. Its advancements amaze the world with new developments and milestones accomplished every week, and its usage spans all the way from business analytic applications, to autonomous driving systems and to smart home gadgets, being one of the most impactful tools ever created [2].

Specifically, Machine Learning (ML) is the application of AI that provides the systems the ability to improve their performance by learning from experience. In this paradigm, Neural Network (NN) algorithms are typically used. Some of these designs have achieved outstanding results in fields like image recognition [3], classification [4], speech recognition [5], data compression [6] or even image generation [7]. For instance, in [8], an artificial neural network surpassed human accuracy in diagnosing lung cancer from images, and quantum properties of an organic molecule have been in predicted in [9]. These are clear signs that both the industry and the research community are really pushing forward to the development of this revolutionary technology, which some argue is driving Humanity to a new era.

Moreover, as can be seen in Figure 1.2, the interest in machine learning has rocketed in the last years, not only in academia but also in the general population.



(a) *Popularity in academia*

(b) *Popularity in general*

Figure 1.2: *Machine Learning popularity*

Even though some of the artificial intelligence techniques that have enabled these shocking results are quite recent [10, 11], the reality is that the vast majority of the theoretical background has been known since the early 60s [12]. However, it was not until these algorithms were really feasible at a hardware level that this technology came into our daily life, a fact that shows the importance of software-hardware codesign.

As shown in [13], generally the progress of the neural networks effectiveness is tightly linked to their size. The quest for neural nets without errors has soared the number of parameters in each layer as well as the number of layers itself, making neural networks greater and deeper every time.

Although it highly depends on each neural network, a larger neural net will generally require a higher computational expense, since more memory will be needed to store all the data, and more computational power will be required to process it. In this situation, machine learning and data analysis experts switched from using CPUs to GPUs for both network training and inference, because they can handle the data of those algorithms way smoothly due to its similarity to image processing techniques.

Nonetheless, the future of artificial intelligence includes edge computing and mass deployment solutions for devices such as self-driving cars or home assistants, which require specialized hardware to get the most out of this technology. While in the cloud data centers seek high speeds, in the edge the devices need to be very power efficient. In order to manage these data-intensive tasks, the research community has introduced the AI accelerators, domain-specific architectures especially designed to meet those expectations and maximize performance. Until recently, most of the developments have been carried out for enhancing training or inference at data centers. However, this trend is expected to shift and edge computing - in smartphones, cameras, sensors, robots and so on - will represent most of the processing part [14], especially for inference. This is not only due to latency reasons, but also due to privacy concerns [15].

As will be discussed in Chapter 5, DNN acceleration is an active area of research and there is still margin to improve it. For instance, most of the designs are communication-bounded or underestimate the impact of the movement of the data. Current accelerators rely on traditional Networks-on-Chip (NoC) to handle their communications, which implies certain limitations.
This thesis proposes an alternative to traditional designs by making use of Wireless Networks-on-Chip (WNoC), an interconnect technology that can overcome those limitations at a reasonable overhead cost. By placing wireless transceivers at chip scale, new opportunities appear. For instance, the most salient advantages of this fabric are the native broadcast support and its flexibility as compared to wired topologies.

The objective of this thesis is to explore the potential benefits of including WNoC technology in hardware acceleration for AI, as well as to design an accelerator with improved performance.

2. Background

In this chapter, relevant information about NN, DNN accelerators, NoCs and WNoCs will be presented to provide certain background.

2.1 Deep Neural Networks

Artificial Neural Networks, or Neural Networks, can be defined as an interconnected group of nodes that simulate our neural system. Each node represents an artificial neuron and each link represents the weight between each neuron. These neurons are grouped in a layered shape, being the neural network a set of these layers. As Figure 2.1 shows, the first layer contains as neurons the raw inputs of the neural net, the last one contains the final outputs and the layers in between are called hidden layers.

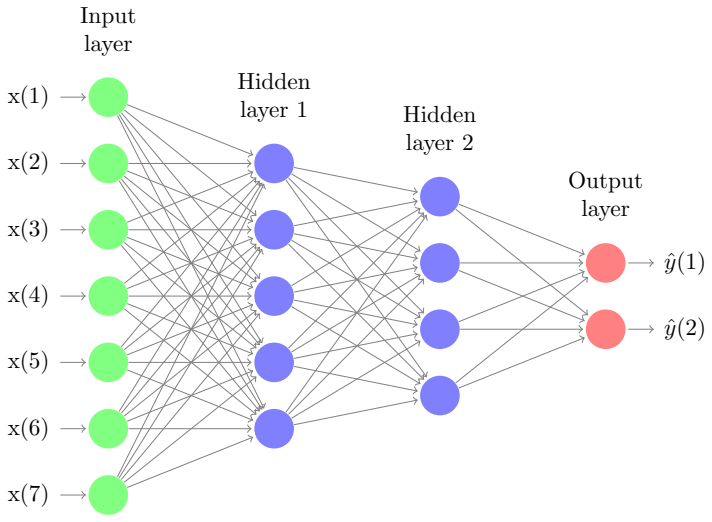


Figure 2.1: *Neural Network example [16]*

For instance, in a NN carrying out image classification, the input layer will be made of the pixels of the image and the output layer will contain the probabilities of each classification class.

The operational principle of Artificial Neural Networks is the following: starting in the input layer, each neuron of the next layer computes its output value by computing a weighted sum between its input vector (each neuron of the input layer) and the weights connecting them. In essence, each weight is deciding the impact or the relevance of the previous neuron to the next one.
Then, a certain value called bias is added to that result. This bias will differ in every node, and can be thought of as a weight that always has a unit input. After that, some layers also use a non-linear activation function, in order to limit the range of working values.
This process is repeated by every neuron at every layer, until we reach the output layer.

In essence, this structure is used to model highly non-linear functions. However, before being able to be used as inference - the process explained earlier - it has to be trained. To do so, iterative algorithms such as backpropagation are applied to tune all the parameters - weights and biases - until it achieves almost error-free results. These algorithms basically propagate the error information backwards in the NN so that each parameter is tuned accordingly.

In Figure 2.1, every neuron is connected to all the neurons in the previous layer. That is, all the layers are Fully-Connected layers (FC) because every link has a non-zero weight. However, state-of-the-art networks are in fact composed of different kind of layers, which include not only FC but also layers in which the neurons are not completely connected to the previous layer. In FC layers, due to dense links, the number of computations in these layers increases quickly with the number of neurons, and become inefficient. Moreover, depending on the application, there are certain patterns in the data that create dependencies not requiring such dense connections.

For instance, in image processing applications, we are interested in learning spatial dependencies because in images nearby pixels tend to be related. In this case, fully connecting all the neurons is excessive and does not perform well, and in these scenarios Convolutional Neural Networks (CNNs) are more suitable. In contrast to fully-connected designs, CNN allows the network to be much more computationally affordable. For example, this kind of networks is widely - but not only- used for object detection.

Other than FC layers, in a CNN we can find layers such as Convolutional layers (CONV), Pooling layers (POOL) and others, as Table 2.1 shows, each of them having a purpose. POOL layers, for instance, are used to reduce the spatial dimensions of the data throughout the network.

In a CNN, however, the most common layer is the CONV layer. For instance, in Figure 2.2 we show the Google Inception v3 CNN, where all the orange boxes represent CONV layers. In addition, they are the ones that take about 90% (e.g. 99.25% in ResNet50 [17]) of the computation and runtime [18, 19].

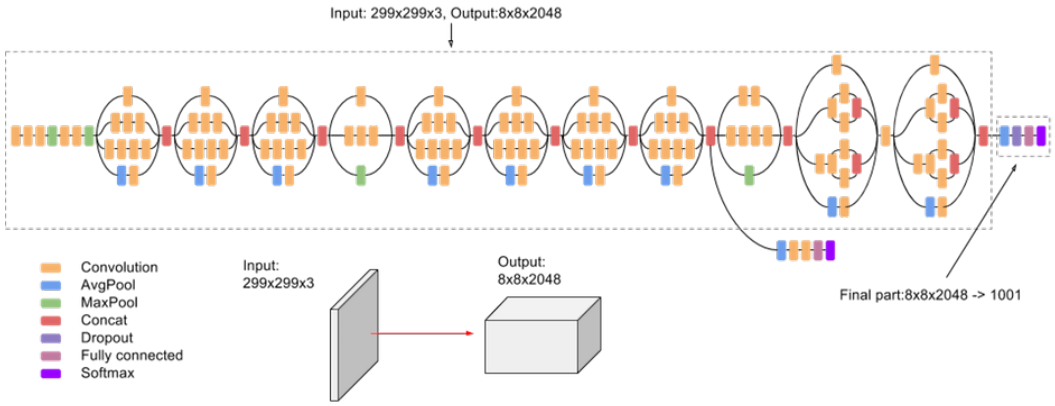


Figure 2.2: *Inception v3 CNN architecture [11]*

In these layers, the inputs, filters and outputs can be expressed as volumes. In essence, a CONV layer computes convolutional operations between the input and a filter, which is spatially smaller than the input but has the same depth or channels. As a result, the output is a smaller volume but its depth is determined by the number of filters. Figure 2.3 allows an easier understanding of this operation: We have M filters and N inputs, both with a depth of channels equal to C. Every filter slides through every input to perform the convolution, creating N outputs with a depth of M.

Since recent NN have a huge number of layers, the nomenclature Deep Neural Networks (DNN) is often used. In these designs, the initial CONV layers extract basic features such as edges and late CONV layers extract more and more abstract features.

Even though some of the work done in this thesis could be extrapolated to other neural networks, the main focus here has been the CNNs and DNNs inference, due to its wide applicability and current relevance.

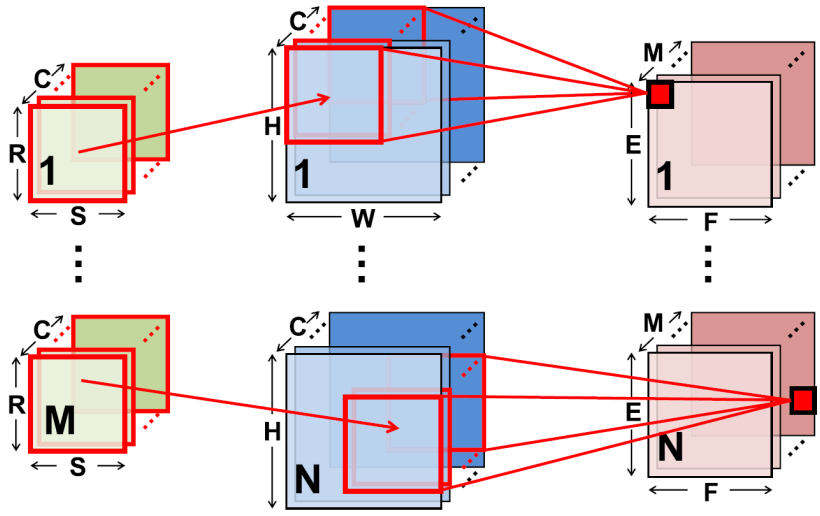


Figure 2.3: *Computation of a CONV layer [18]*

DNN	Year	Num CNV	Num RNN	Num POOL	Num FC	Filter Sizes	Input Sizes
Alexnet	2013	6	0	1	1	11x11, 5x5, 3x3	224x224
Googlenet	2014	59	0	16	5	1x1, 3x3, 5x5	224x224
Resnet-50	2014	49	0	2	0	1x1, 3x3	224x224
VGGnet-16	2015	13	0	5	3	1x1, 3x3	224x224
DeepSpeech2	2016	2	7	0	1	41x11, 21x11	13x41x11
Deep voice	2017	0	40	0	3	-	28x29

Table 2.1: *Parameters of recent DNN [20]*

Having said that, this work does not focus on neural networks development or analysis at algorithmic level, and therefore a further explanation is not included here. For detailed information in the topic, related literature [21] is suggested.

2.2 DNN accelerators

Historically, computer systems have included specialized architectures to assist the CPU in specific tasks. A great example of this are the GPUs (Graphics Processing Units), which can handle images much more smoothly due to their specialized architecture.

In a broad, general sense, systems tend to follow the behavior that Figure 2.4 depicts: flexible systems such as CPUs are not optimized for any particular application and are consequently slower than dedicated architectures such as an Application-Specific Integrated Circuit (ASIC), which is the most rigid one. Other systems such as Field-Programmable Gate Arrays (FPGAs) achieve better flexibility by being programmable, becoming in general more efficient than a GPU but slower than an ASIC.

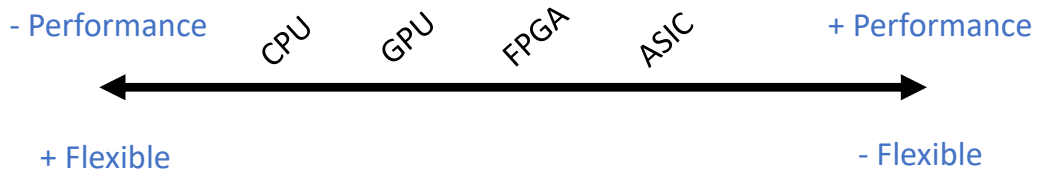


Figure 2.4: *Flexibility vs Performance*

In the AI field, data scientists switched from using CPUs to GPUs due to the similarity between neural networks and image processing techniques, which increased the performance in demanding tasks such as training [22] mainly because of the great parallelism that GPUs can offer. However, the future of ML requires specific architectures capable of handling billions of parameters in a more energy-efficient way: the DNN accelerators.
While GPUs achieve efficiencies of 0.1 Tera Operations per second per Watt (TOPS/W), ASIC-based hardware designs can reach 1-10 TOPS/W. Moreover, when these operations are to be computed at the edge level - embedded in sensors, smartphones or cars - we have strict power envelopes. Therefore, as opposite of what happens in the cloud, here we need much more specific hardware.

Examples of state-of-the-art DNN accelerators are the Google TPU[1] [23], Eyeriss [18], and Intel Nervana [24]. Figure 2.8 shows a comparison between CPU, GPU and TPU.

These accelerators are mainly specialized in computing discrete convolution operations, since it is the most common operation to be solved in DNNs. These operations can be then broken down in several multiply and accumulate (MAC) operations, as shown in Equations (2.1) and (2.2).

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.1}$$

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n), \tag{2.2}$$

where S is the output, I represents the input and K is the filter.

The idea behind DNN accelerators is to take advantage of specific data reuse and parallelism opportunities that happen in DNNs. For instance, when a convolution operation is carried out, its sliding nature enables the possibility to use the same value (e.g. a certain weight in a filter) multiple times for different input values. Moreover, since some of the dimensions in these operations are separable, we can parallelize certain independent calculations to save time.
Another broader way of seeing it is using the matrix-vector notation for each layer in neural networks, where the matrix $\boldsymbol{W}$ contains the weight values $w_{i,j}$ from neuron j of the previous layer to neuron i of the next one, and the column vector $\boldsymbol{x}$ contains each activation in the first layer.
Then, the output can be computed as $\boldsymbol{y} = \boldsymbol{W} \cdot \boldsymbol{x}$. To calculate the result, an accelerator can efficiently reuse the values in $\boldsymbol{x}$ to iterate the dot product in each row of the weight matrix, instead of wasting resources accessing the same data repeatedly. For instance, Figure 2.5 shows how an accelerator can partition the data to speed up the computations.

[1] Google TPU accelerator is based on a Systolic Array design. In these designs, the data is distributed neighbor-to-neighbor, being the analogy to sliding windows and maximum reuse their major strengths.

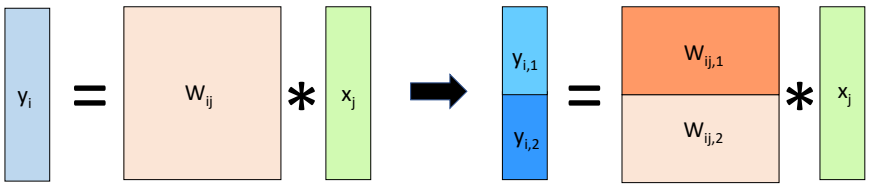


Figure 2.5: *Matrix-vector product in an accelerator*

The hardware architecture of an accelerator can be generalized as an off-chip memory and multiple chiplets accessing to it. Each chiplet has a shared memory and an array of several Processing Elements (PE) interconnected using a NoC to allow parallelization. Each PE has at least one MAC unit, responsible of the most fine-grained operation, and can have a small memory as well.

The general operational principle of a DNN accelerator to compute a layer output is the following one: The first stage is the distribution, when the weights and inputs are sent to the PEs through at least one NoC. The second step is the computation, when the partial sums are obtained by the MAC units. Finally, a gather stage adds up the values accordingly and the memory collects back the final outputs.

These three steps can be followed strictly separated, as most of the accelerators do, or in a pipelined and orchestrated manner, such as new accelerators suggest [20].

During this process certain data can be reused, which we can split it in three main categories, as Figure 2.6 shows:

- **Spatial reuse:** When different PEs need the same data in the same iteration, multicasting opportunities appear. It requires multicasting or broadcasting support.
- **Temporal reuse:** When the same PE holds a value during multiple iterations, it can be reused for multiple operations. It requires buffers at PE level.
- **Spatio-temporal reuse:** A value can be forwarded from one core to another to be reused in the future, instead of reading new data from memory. It requires forwarding links between PEs.

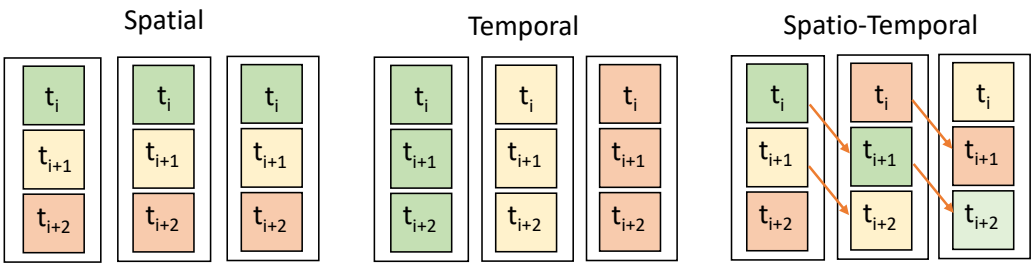


Figure 2.6: *Data reuse in an accelerator*

Another simple proof that these reuse opportunities exist is that while a neural network can have millions of parameters, there are billions of operations to be computed until the final output is obtained. The different ways to map the data into the PEs are called *dataflows*, and they will define how these reuse opportunities are tackled.

When used properly, these forms of reuse allow DNN accelerators to save memory and bandwidth, which in turn saves runtime, area and power consumption. To note this, Figure 2.7 shows the normalized energy cost of accessing the different memory levels in an accelerator. Fetching the data repeatedly from memory is much more expensive than reusing it, and this is why so many accelerators focus on saving accesses to distant memories.

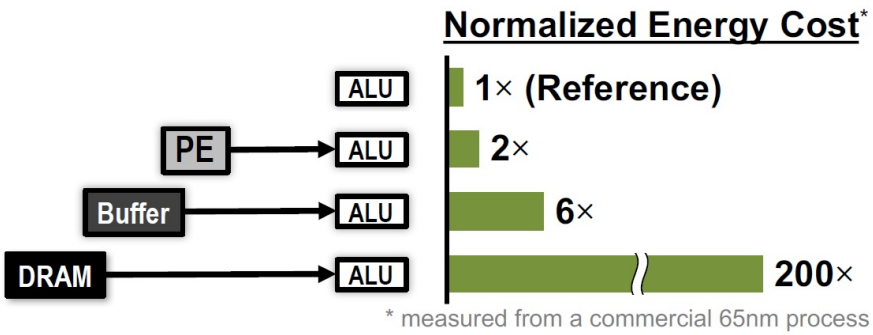


Figure 2.7: *Energy cost of memory access [1]*

Another interesting point DNN accelerators are starting to take advantage of is the *sparsity* in pruned neural networks [25]. This technique allows to remove insignificant links in neural networks, due to its low weight value, reducing the number of parameters. It can be useful not only from a hardware perspective, where storage and computing savings can be made, but also from a software point of view to avoid problems such as overfitting [26]. Moreover, after using the activation functions, some outputs will also have a value near to zero which we can avoid to compute in the next layer.
Logically, hardware accelerators designed to identify and handle such sparse networks can highly benefit from it, something that a generic processor could not do.

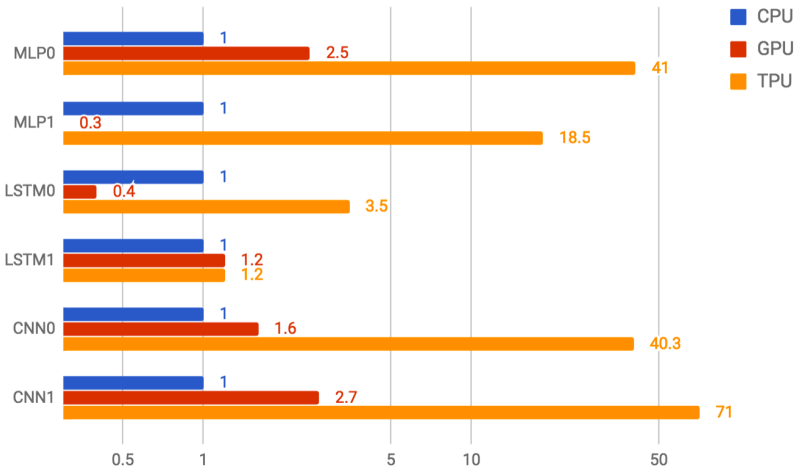


Figure 2.8: *CPU, GPU and TPU performance on six reference workloads [23]*

2.3 Networks-on-Chip

In order to intercommunicate the different components of the die in a DNN accelerator, current systems rely on metal planar interconnects.

The bus is one of the simplest interconnects. It allows low latency and broadcast links due to its shared-medium nature, but when the number of leaves increase, it does not scale and saturates [27]. Moreover, the provided bandwidth is quite low and the power requirement is considerable when the bus is long, mainly due to the need of an arbiter. Hence, networking methods applied to interconnects led to the introduction of NoCs [28, 29]. This methodology allows integrating multiple cores on a die solving some of the bus problems.
NoCs are packet-switched networks that use routers and different protocols to communicate the different nodes, and there are multiple topologies available. The most used ones in these specific designs, apart from buses, are tree-based topologies such as fat-tree, and other options include meshes, crossbars or hierarchical NoCs [20].

Crossbars and meshes can provide better bandwidth and low delays. While the former one is based on switches, the latter uses routers to drive traffic. Additionally, tree-based designs provide non-blocking communications. However, in manycore systems their area and energy consumption do not scale well with the number of cores either [28]. When scaling up, most accelerators use hierarchical NoCs [30], but then multi-hop and latency problems appear, as well as huge power consumption and underutilization at PE level. Moreover, the multicast support is often hard to get when scaling up.

In manycore systems, the number of cores is increasing, in part due to the miniaturization of the

components. This allowed embodying a large number of cores, thousands of them in some cases, in a die, which can effectively increase the computational throughput. By having more cores, we can better parallelize the computations, which means that we can solve the operations faster in different processors, but the communication requirements grow. If the network can provide such requirements, the total throughput can be improved by increasing the number of cores. However, when the network saturates the throughput can not keep improving. This has led to the point where systems are starting to be communication-bounded instead of computational-bounded [31], as Fig 2.9 shows. The capacity of the network to route the traffic generated by the computation elements will change the point where we shift from computation-bounded to communication-bounded situations, and this is why it is so important that the networks scale. This point is usually set aside in evaluation analysis and designs, with the main focus set in the computation and not the communication.

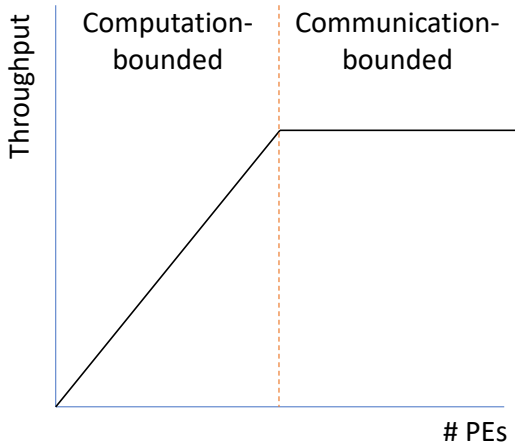


Figure 2.9: *Boundaries in a manycore system*

Moreover, according to the International Technology Roadmap for Semiconductors (ITRS), new interconnect ideas are needed since the metal wires will not be able to satisfy the performance requirements in the near future [32].

2.4 Wireless Networks-on-Chip

To solve all these issues in general manycore systems, the research community has introduced a new type of fabric: the Wireless Networks-on-Chip [33].

In a broad sense, the concept of WNoCs consists in augmenting the cores with wireless transceivers and antennas, which can modulate a signal and send it to other cores within range, usually limited to the package as Figure 2.10 shows. Even though this technology is still in development stage at this

small scale, it already shows great potential.

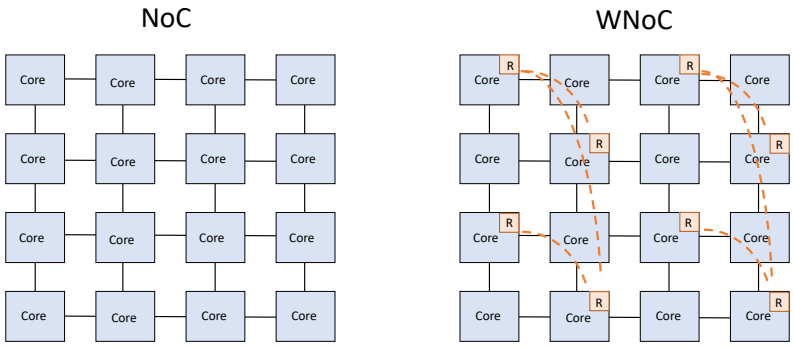


Figure 2.10: *NoC and WNoC general schemes*

For the transmitter and receiver, the commonly used antennas include patch antennas [34], monopoles [35] or zig-zag antennas [36]. Its size is inversely proportional to the frequency band of the transceiver, which can be a favorable point in the future due to the recent research in THz bands [37]. On-chip antennas tuned to the millimeter wave range have been proved to achieve tens of GHz of bandwidth. Antennas working at optical and THz frequencies have been theoretically investigated, showing that these could achieve bandwidths of hundreds of GHz while being proportionally smaller [38, 39]. In this work, a comprehensive study of the antenna possibilities will not be carried out, and a monopole antenna will be assumed as will be explained later in Chapter 6.

Talking about the intra-chip channel, it should be mentioned that it is quite unique in certain terms. For instance, it is quasi-deterministic, because we will not have moving parts, which practically removes the fadings and makes it time-invariant. Moreover, package enclosure confines electromagnetic propagation, which can improve its security. However, it is not easy to model and it is largely unknown, for instance due to the presence of multipath or challenging package modelling. Channel characterization studies, including path loss and delay spread, have been carried out in [40–42].

The main advantages of the WNoC technology can be summarized as follows:

Broadcast support:

Due to the nature of wave propagation, and assuming omnidirectional on-chip antennas, the WNoC provides native broadcast support. This allows multiple and fast communication links at the same time, whereas in traditional NoCs the multicasting and broadcasting often means that the data has to be replicated or accessed multiple times from memory. In a sense, it is as if the wireless channel replicates naturally the data, and we only need to spend resources by adding receivers to increase the bandwidth [43]. In manycore designs, not having features like multicasting support has led to avoid such behavior, or breaking it down into multiple unicasts [44] which leads to unnecessary overheads.

Latency:

Since the data is modulated and sent through electromagnetic waves within the package, it can travel at a speed close to that of light. This ensures an (almost) distance independent communication in terms of delay, granting a reduced latency level. In addition, this topology does not require routing or multi-hop links from the sender to the receiver, saving time and intermediary circuitry such as switches. This not only helps reducing the overall latency, but also facilitates the layout task. The allocation of the cores is not so critical anymore, and thus we have more room to place other wired components whose position can affect the performance severely.

Moreover, the short communication ranges for wireless in WNoCs grants a good signal-to-noise ratio (SNR) due to limited path loss and almost deterministic channel [40].

Flexibility:

Another advantage of WNoCs is the flexibility they can provide. When a transmitter is sending data through the channel, each receiver can decide at run-time whether to process the information or not. This means that this technology allows to virtually map different topologies at every cycle. On the contrary, wired NoCs are fixed and very few seek to achieve flexible data movements. This limits substantially the performance of manycore systems when irregular data traffic has to be routed.

Additionally, since the transmitted data will have to be serialized, the WNoC can also offer precision adaptability between layers as well as error flexibility by changing the wireless bit error ratio (BER), allowing the whole system to save power when less accuracy is required. In neural network algorithms, this has been widely studied [45], concluding that we can use different precision length at different layers to save energy and still achieve lossless accuracy results. If a similar approach were to be pursued in a wired system, we would have to design the architecture to work in the worst-case, that is maximum precision, wasting resources when such level of precision is not needed.

Another possible way we could save power is by reducing the speed of the transmission. For instance, when broadcastable data - instead of K unicasts - needs to be sent in a cycle, we could theoretically transmit around K times less power because we can increase by K the bit time and the bit energy would remain the same. However, it would require some adaptability, and thus complexity, in the receiver.

Moreover, other well known techniques at large scale wireless links could be applied. In terms of multiple access to the shared medium, multiple channels have been proposed in [46] by implementing transceivers at different frequency bands. Beamforming, which allows spatial multiplexing, has been studied in [47,48]. It can enable multiple simultaneously communications within package by narrowing the antennas beam and directing the data to specific points in the chip. CDMA techniques have also

been tested [49]. Yet, the drawback for all these large-scale options is the increased complexity of the transceivers, which affects the area and power consumption.

Scalability:

In a higher level sense, when NoCs try to scale they face problems like exponentially growing areas and power consumption, multi-hop problems or even layout difficulties [50]. In this scenario, the WNoC systems scale linearly when the number of cores increases. For instance, in a tree, when the number of leaves increases, the whole network has to be changed and exponentially augmented to feed all the cores, including switches and metal interconnects through all the NoC. In a bus, not only the metal interconnect itself but also the arbiter complexity increases exponentially. In a WNoC, the wireless nature allows us to just include an extra transceiver at the core, no matter where it is located in the package.

In addition, wireless energy efficiency generally scales much better with the distance as well. As can be seen in Figure 2.11, when distance between communication cores increases in manycore systems, wireless interconnects can be advantageous.

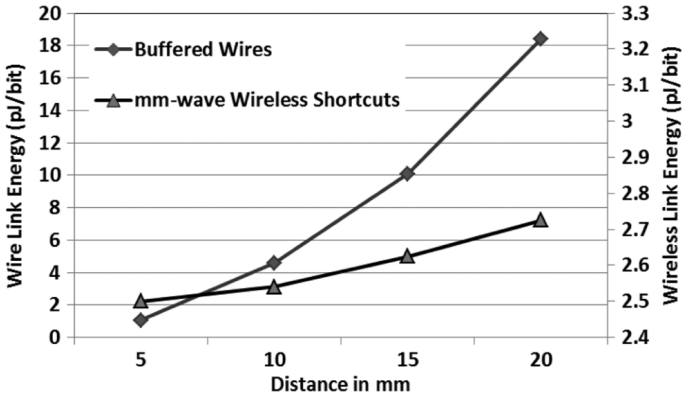


Figure 2.11: *Variation of energy dissipation per bit with distance for a wired and mm-wave wireless link [51]*

All these appealing features, however, come at an area and power overhead and limited datarates, which will be studied later in Chapter 4. Moreover, the shared-medium nature of the WNoC requires protocols like MAC and synchronization in maycore systems.

It should be mentioned that when comparing a WNoC to a traditional NoC in terms of area, power consumption or datarates alone, the NoCs will likely perform better than the WNoC, especially in systems with few cores. In terms of bandwidth, for instance, WNoC are still limited comparing it to wired Networks-on-Chip and they do not scale so well. However, by taking advantage of the above mentioned features, certain improvement can be found, as we will see.

3. Motivation

This section analyzes the behavior of the data movement in DNN accelerators in an attempt to motivate the suitability of WNoC within the context of DNN accelerators.

3.1 Dataflows

As already mentioned, the distribute-compute-gather process in DNN accelerators happens repeatedly following a loop structure. However, that does not mean that the data is always flowing in the same order or direction.

To think about the different options we have when mapping this data, we can represent the convolution operation as a nested loop, as Figure 3.1 shows. Changes in the loop order will not affect the output result but will impact the hardware that runs the algorithm. We use a nomenclature based on letters to refer to the different dimensions of the data, as can be seen in Figure 3.2: C: input or weight channel, K: number of filter or number of output channel, N: number of input, R/X: row of the filter/input, S/Y: column of the filter/input, and X' and Y' represent output row and columns. Some of these dimensions are coupled within the type of data. For instance, the channel size should be the same in the input and in the weights, and this number of channels will become the output depth. Additionally, X' and Y' are fixed once the other dimensions are given. Therefore, the dimensions set a 7D space.

```
for(n=0; n<2; n++)
 for(k=0; k<4; k++)
  for(c=0; c<6; c++)
   for(y=0; y<8; y++)
    for(x=0; x<8; x++)
     for(r=0; r<3; r++)
      for(s=0; s<3; s++)
       O[k][y-r][x-s] += W[k][c][r][s] * I[c][y][x];
```

Figure 3.1: *Loop nest representation [52]*

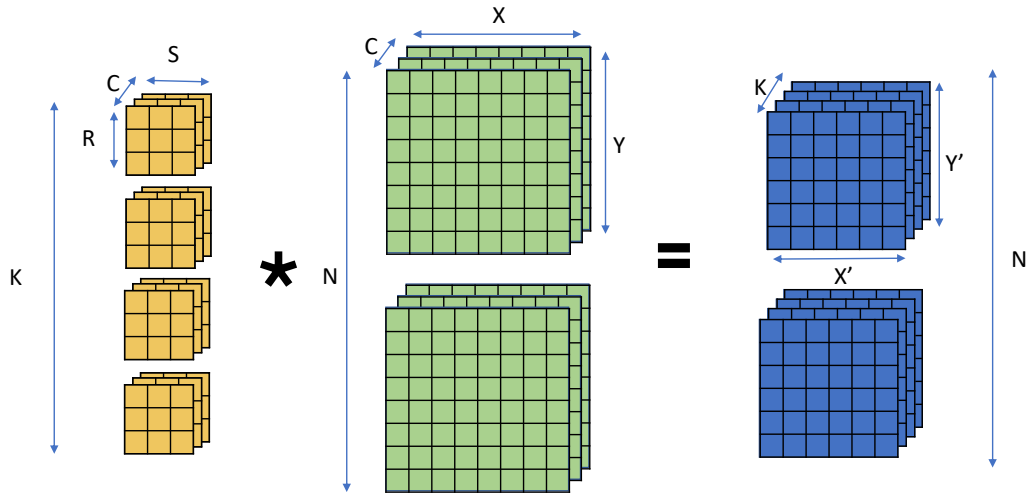


Figure 3.2: *7D representation*

In a simplified way[1], an accelerator maps down every layer in order to its PEs, starting by the first layer, computing the output activation and using that as the input for the next layer. The mapping strategies to send the weights and inputs to each PE, like its order and size, have several degrees of freedom and its design is not a trivial task. For instance, one way to do it could be to make the PEs fetch certain weight and input values at every cycle, compute the partial sum and send it back. However, by doing it this way we are not taking advantage of the reuse opportunities we already talked about.

Another smarter way to do it is to make the PE hold a weight value until all the calculations involving it are done, reducing the movement of data by only fetching new inputs. These different mapping strategies are called *dataflows*, and their exploration is an active research topic due to the huge space it represents and the impact in performance and broadcast opportunities that it produces. In a conceptual sense, they can be classified as:

- **No Local Reuse (NLR):** Each PE fetches both input and weight values at every step. Even though it does not reuse the data, it has other advantages like unneeded local storage at PE level.
- **Weight Stationary (WS):** The PEs keep the weight value and the inputs change at every cycle. This can be understood as if the inputs were sliding through the filters. In the loop notation, the dimensions corresponding to weights are in the outer loops.

[1]Some accelerators also take advantage of cross-layer reuse [53], which is a complex process and beyond this report.

- **Input Stationary (IS):** The inputs are held in the PE until all the computations involving it are done. The weights slide through the inputs in this case.
- **Row Stationary (RS):** An entire row of weights and inputs is mapped in the PE, breaking the 2D convolution down into parallel 1D convolutions, which will be reduced afterwards [18].
- **Output Stationary (OS):** Each PE is responsible to output a final single pixel. This means that all the partial sums are added up at the PE, and there is input reuse among neighbor PEs. The output activations are in the outer loop.

Equations (2.1) and (2.2) represent the same convolution operation with different sliding classes, just like IS or WS. However, far from representing single implementations, these only state the concept behind them, because several different dataflows can be considered WS, for example. The goal of a good dataflow, in a broad sense, is to take advantage of the algorithmic data reuse to obtain hardware reuse.

In this scenario, the mapping of the nested loop we previously talked about is IS-like, since the inner-most dimension, the one that changes the fastest within the loop, corresponds to the weights, assuming that we do not map entire filters to the PEs[2]. A way to see the stationarity in the dataflows is to identify the dimension that changes and not over the most fine-grained tile updates (the inner-most temporal-for). That is, we should find the dimensions that do not change the fastest.

Figures 3.3 and 3.4 show the data distribution when mapping down two examples of the most representative dataflows, WS and IS, into PEs. The actual mappings vary depending on the dimension size, order and other factors, and therefore this is just an illustrative example. In this weight stationary dataflow, each PE fetches a different filter and holds it for the required number of cycles. In addition, all the PEs fetch the same batch of input data, which will be broadcasted. At the next interval time, another batch of input data will be broadcasted, and the filters will remain. As a result, every PE is computing a different channel of the output, and different time steps output different output batches. The first step outputs all the channels of the first output, the second step outputs all the channels of the second output, and so on.

On the contrary, in the input stationary dataflow the filters will be broadcasted for all the PEs to fetch it at every time step, whereas different batches of the input will be sent to different PEs. In this case, each PE is computing the same channel from a different output and the N outputs are being computed at once. The first time step outputs the first channel for all the outputs, the second one outputs the second channel for all the outputs, and so on.

[2]However, we would actually need information about the hardware parameters and the neural network shape to define it as an IS dataflow, as Chapter 4 explains.

For the sake of simplicity, the figures only show a limited number of filters and inputs, and specific dataflows, but the idea can be extrapolated to any number of inputs and filters, and most of the dataflows and sizes.

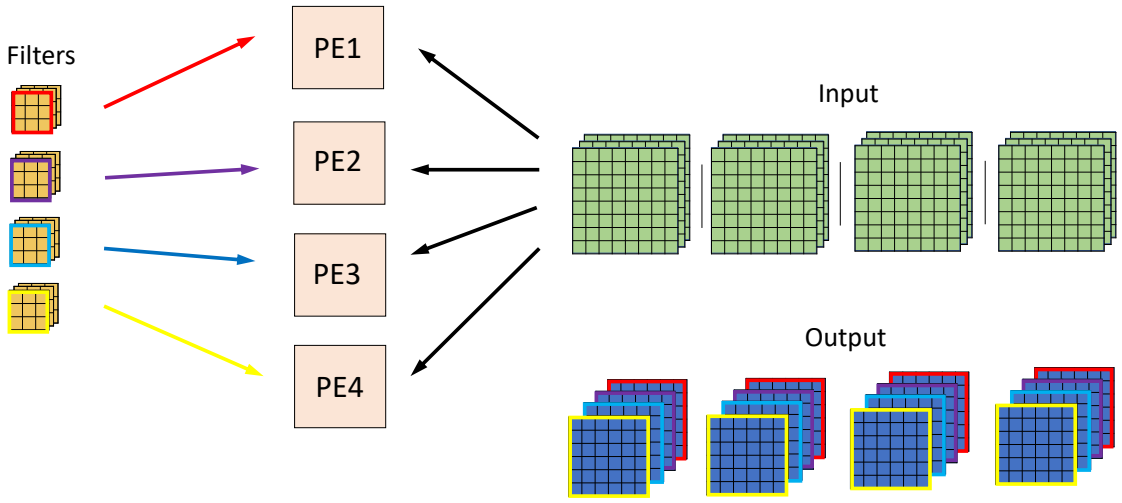


Figure 3.3: *Weight stationary dataflow mapping*

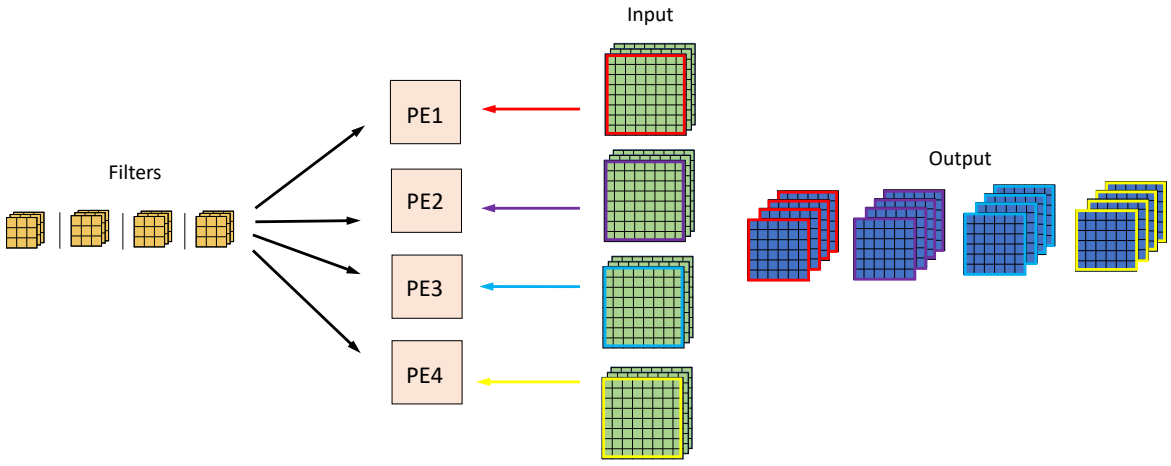
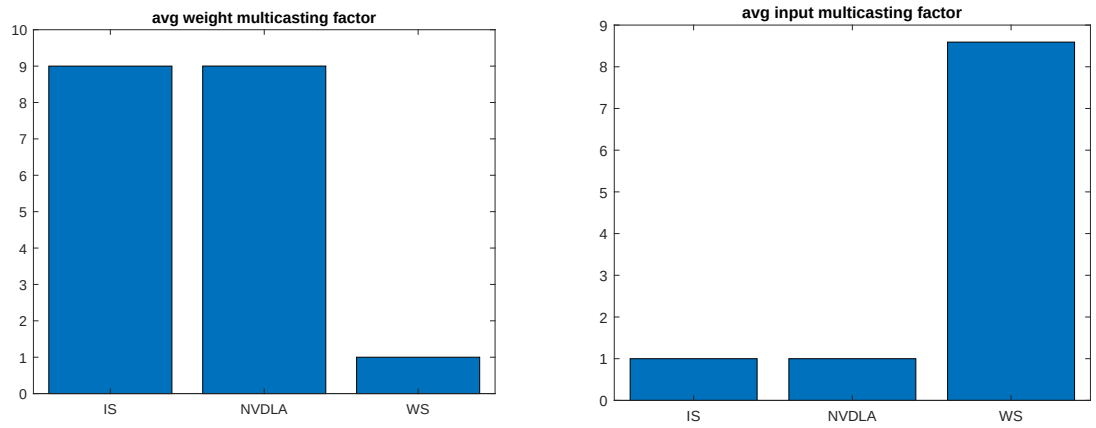


Figure 3.4: *Input stationary dataflow mapping*

As can be seen, several broadcast opportunities appear. Even though it highly depends on the hardware parameters and the specific dataflow implementations, in the case of the simple WS dataflow it tends to allow the broadcast of the input values at every cycle, while the IS case allows us to broadcast the weight values. Figure 3.5 shows the multicasting opportunities at a cycle level for different dataflows, including as well a NVDLA dataflow, which follows a mapping strategy similar as to the input stationary one. This multicast factor is an indicator of how many times the same datum is reused in a PE.



(a) *Weight multicast*

(b) *Input multicast*

Figure 3.5: *Average multicast factor*

We previously said that the dataflow exploration represents a huge space to go over. The dataflow choice sets the order in which the different data classes will iterate in the nested-loop. Apart from the order, we can also change the tiling size, that is how many data we send to each PE to be computed, and the stride, meaning the steps taken between each data sent at each PE.
In addition, PEs can spatially map (over the PE array) or temporally map (over different iterations) distinct dimensions of the neural network.

All these options lead to millions of possible dataflows when it comes to map a neural network to the hardware, and the dataflow choice has an important impact in the performance of the accelerator. From all these possibilities, only a tiny amount of them are optimal, and it should be the aim of the designer to find them.

This exploration problem would be relatively solvable if once we found the optimal dataflow we could stick to it during the entire neural network inference. However, after studying the behavior of different dataflows at different layers, it is obvious that the different layers in a neural net will notice the impact of the dataflow used, as it is deductible from the study carried out in Figure 3.6, and as it has been discussed in [54]. This is logical, since each layer will have different sizes of its parameters, and therefore different parallelization needs. For instance, in typical CNNs the early CONV layers tend to get bigger inputs than the late layers, because the former ones extract simple features directly from the activation data. Then, a WS dataflow will usually be useful in the first layers since more input values will be able to be broadcasted into the PEs. Similarly, late layers perform way better when using an input stationary based dataflow, since the filters represent a bigger computational cost there. However, this does not represent an exhaustive dataflow exploration and therefore these two are not the optimal ones for these layers but a mere representation about the impact of its variability. There is a different

dataflow, with different temporal or spatial mappings, tiling sizes and order permutations that would give us the best performance in each layer. Nevertheless, most of the state-of-the-art designs usually use the same dataflow statically [55].

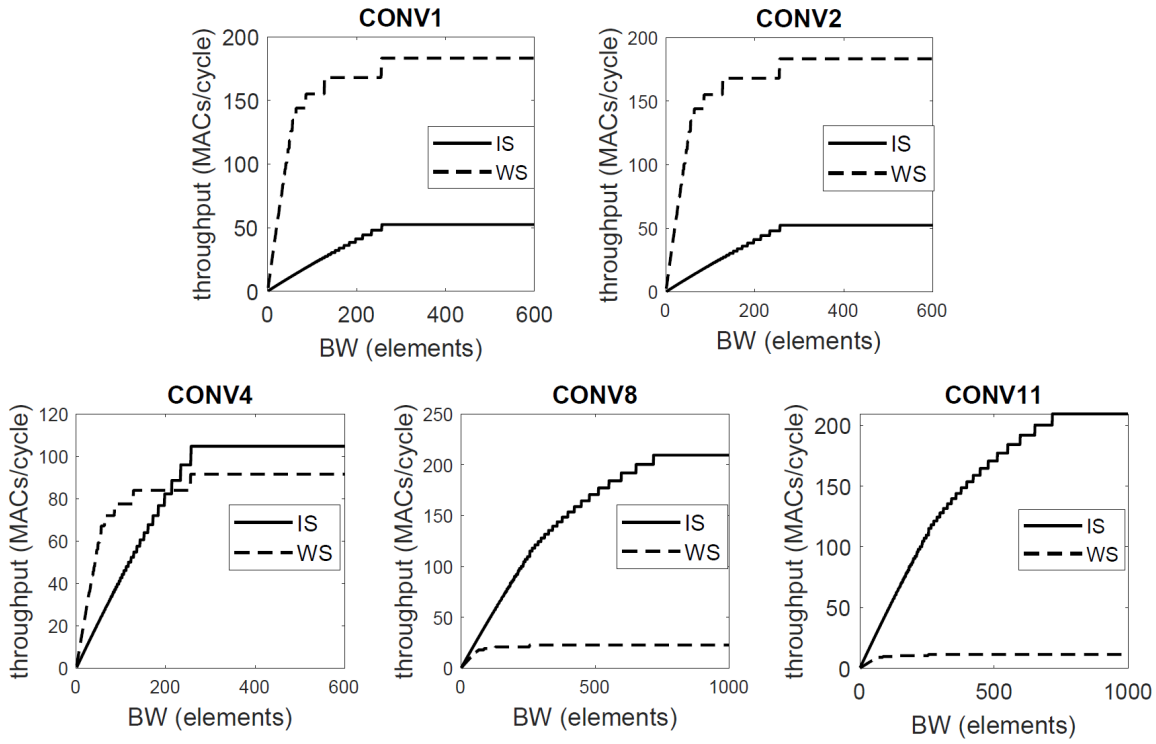


Figure 3.6: *Throughput analysis for WS-like and IS-like dataflows in VGG16 neural net*

Consequently, different layers require different dataflows for the hardware to be efficient when computed, which in turn means that the NoC needs to adapt to these changes between layers and support its traffic.

An interesting point to look at in Figure 3.6 is the CONV4 layer. A surpassing point around a bandwidth of 225 elements can be found, indicating that for low bandwidth resources the WS-like dataflow should be chosen but the IS-like dataflow would perform better at higher bandwidths.

Until this point, and for motivation purposes, we only talked about dataflows at PE level, but we can also cluster different PEs to create another level above them. In this case, the dataflow space is even bigger and the community still lacks a tool to efficiently explore the optimal one. A further detailed explanation of clustered dataflows will be done in Chapter 4.

3.2 WNoC in DNN accelerators

As introduced, accurate NNs tend to be bigger every time, which include more parameters and raise more parallelization opportunities when scaling. Since the goal of DNN accelerators is to parallelize its computations, the number of PEs can be quite large, reaching thousands of them in some cases. This makes DNN accelerators a manycore-like environment, becoming similar to the architectures the WNoC are made for. Therefore, it seems appropiate to explore the match between these two technologies.

In the previous section we have seen how broadcast opportunities appear and can benefit the throughput of the entire system. Therefore, to be efficient, DNN accelerators require one-to-all support. In these architectures, the multicast traffic will be of utmost importance in the distribution step we mentioned early, when the data is being moved from the memory to the PEs, but not so relevant in the gather step, where no one-to-all traffic appears. In a traditional NoC, in which the designs currently depend on, this broadcast is hard to do, while in a WNoC we have inherent support for it.

Another matching point is the flexibility need of the design to adapt to the flow of the data.
We have seen how there is not a unique dataflow that is optimum for every existing layer. At hardware level, the different dataflow mappings would usually need complex NoC tolopogies to be mapped in different ways. To change the routing of the different classes of data at real-time is hard to achieve in traditional wired NoCs, which tend to be static, but WNoCs are naturally good at it. As argued, different layers will need different batches of the data to be broadcasted or unicasted. While fixed wired topologies do a really good job in terms of bandwidth and area and power consumption when specializing in each of those, they are usually not adaptive. Using WNoCs, we can potentially map the different multicasting or unicasting routes easily, that is we can create dynamic connections on-demand, and adapt the consumption to it. Moreover, as can be seen in Table 2.1, this short list of neural networks already depicts a wide variability in terms of number of layers and sizes of filters and inputs. In this case, the WNoC flexibility not only is useful for inter-layer adaptability but also between neural networks. For instance, in traffic situations like broadcast, the wireless bandwidth requirement suddenly drops, and the transceiver can save power for some time slots (not area, which is fixed). To understand this, we have to talk about the wired bandwidth and the wireless bandwidth.

In a wired perspective, we talk about bandwidth as the number of values, in bytes, we can send from one point of the package to another point of the package, in a single cycle. In a wireless perspective, however, we usually talk about the bandwidth in terms of Hertz, which will provide us a certain datarate, in bits per second (bps), depending on the spectrum efficiency. Therefore, since the wireless data will be serialized in every cycle, the needed datarate can be found as $R_b(bps) = BW(bytes) \cdot f_{clk} \cdot$

$8(bits/byte)$, where $R_b(bps)$ is the datarate, the $BW(bytes)$ is the wired bandwidth and the f_{clk} is the clock frequency of the accelerator.
In this sense, when broadcasting a certain weight, the wired topology will have to replicate the value to reach all the cores in the network even though the number of unique elements is 1. However, the wireless datarate will be low and the transceiver can save power[3]. For instance, the wireless NoC only needs 1.6 Gbps to broadcast 8 bits to N receivers at 200 MHz, whereas certain electrical topologies would need up to 8N wires in the worst case. Nevertheless, it should be mentioned that architectures like [20] address this flexibility problem by using switches and controllers, which still grow exponentially with the number of cores.

Latency is another point in which WNoC can be useful for DNN acceleratots. When a PE has to wait for some data to reach him, we are underutilizing that resource and it is a scenario we should avoid as much as possible. This problem is called starvation and it can be caused by NoC saturation due to multi-hop routing or lack of bandwidth, mainly off-chip. In WNoC, the multi-hop and latency problem can be easily avoided, as mentioned earlier, by providing long range single-hop links.

Since a DNN accelerator is a multi-chip environment, accessing to an off-chip memory can be challenging with the number of cores increasing. As will be tackled later in Chapter 5, WNoC can also alleviate these situations by providing high bandwidth and broadcast friendly support, in contrast to some of the fabrics that lead to communication blockings.

One of the drawbacks of WNoC is the need to deploy medium-shared protocols. In a DNN accelerator, however, we can avoid that if using WNoC only from memory to PEs, that is in a distribution manner. In this case, there is only one transmitter at a time and each PE can choose when to listen the medium[4]. We thus avoid the need of collision avoidance or detection protocols, which would be needed in a general manycore architecture with WNoC.

However, we have to take into account that the vast majority of the WNoC designs until this date have been proposed for general manycore systems. In a ML accelerator, since every circuitry is very specific and optimized for a certain task, the overheads of including wireless transceivers may be greater than in a more common manycore system. Nevertheless, flexibility, low latency and ease to broadcast have been the major motivations for scaling accelerators by merging them with WNoC technology.
Given the predicted limitations of traditional NoCs and the expected growth in specific ML hardware architectures and WNoC technology [44], the sole idea of implementing WNoC transceivers in accelerators has already value by itself, even if as today it were not feasible or advantageous yet.

[3]For feasibility, we can keep the same datarate but only be active for one time slot, saving energy [56].
[4]The assumption that the receivers know when to listen to the medium requires some controller logic, such as wired logic or wake-up radio [57], which is not included here.

4. Methodology

In this chapter, the used tools and essential proceeding information will be presented.

4.1 MAESTRO and Data Mappings

To run the experiments for the dataflows, Synergy Lab tools have been used. The most used one has been the MAESTRO software [52], an open-source analytical tool written in C++ that models dataflow mappings into DNN accelerators and evaluates its performance. Since it is based in analytical models, it is not a cycle-accurate tool, meaning that instead of running real-time simulations and traffic patterns, it computes the performance of the dataflow under some hardware resources when mapping a certain layer by estimating bandwidth and memory requirements, delays and other parameters. It has been validated by showing an average error rate of 5.9% as compared to a cycle-accurate analysis in the MAERI [20] accelerator.

MAESTRO gets as an input a complete description of the hardware resources, dataflow and layer parameters, as Figure 4.1 shows, and outputs computational throughput (MACs/cycle), runtime (cycles), area and power estimates, among others, for each layer.

The two main instructions in MAESTRO are *TemporalMapping (A,B) d*, and *SpatialMapping (A,B) d*. The former one depicts the changes in time iterations in the PEs, when mapping A values of the dimension d with a step B. The latter describes spatial distribution of a tiling size of A values for the dimension d within an array of PEs, with a step B. The order of these instructions will define the loop-nest position for every dimension and is closely related with the temporal and spatial reuse. Figures 4.3 and 4.4 may be helpful to understand it.

To run its analysis, MAESTRO calculates both computation and communication delays as well as hardware area and power consumption. It assumes a bus-based NoC at 28nm technology, and includes an arbiter.

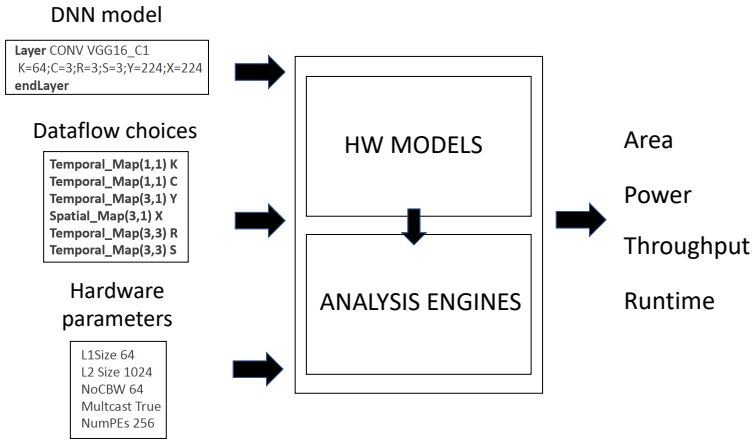


Figure 4.1: *MAESTRO scheme*

As an example of the usefulness of this software, we plotted in Figure 4.2 the results of some iterations for number of PEs and NoC bandwidth, which will vary the area of the total system. After cleaning and organizing the data points, this plot could, for instance, help to design an accelerator by targeting a throughput and finding the parameters that minimize the area for a specific dataflow.

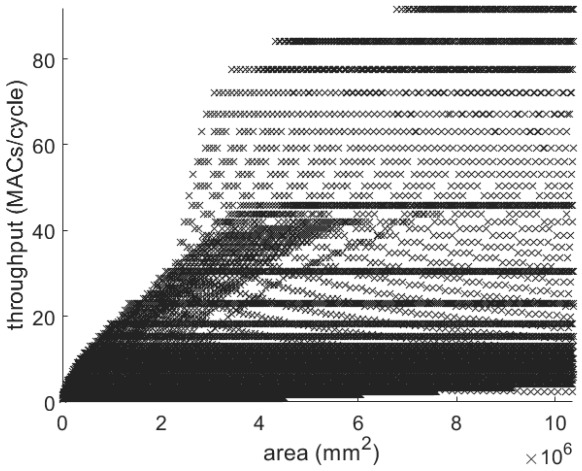


Figure 4.2: *Throughput as a function of the area example*

This work helped in a small part to find bugs in MAESTRO, as well as to enhance it. The current MAESTRO version has been augmented with the WNoC transceivers models for area and power, which allows the software to compute the whole consumption estimation for both NoC-based topology and WNoC-based topology.

Returning to the accelerators, to create realistic and scalable dataflows in practice, PEs are grouped together in clusters. This clustering opens a whole new spectrum of dataflow possibilities, since now

the spatial mapping dimension in the inner level, that is the PE level, can be distinct from the spatial mapping dimension in the clusters level, the outer one. In MAESTRO, we can define as many cluster levels as desired. When mapping the data into clustered PEs, the tiling size has an important impact. For instance, we can create situations in which the PEs in each cluster can compute the whole dimension in a single iteration, becoming *unrolled*. On the contrary, if the tiling size is big, the PEs will have to iterate some values inside a cluster, allowing latency hiding, and we can then see different dataflow behaviours in each level. For example, we could observe an input stationary dataflow inside the cluster across PEs, but weight stationary behavior from a cluster point of view. Some of these examples can be found in the Annexes.

In this work, several dataflows and tiling sizes have been evaluated and compared both for clustered versions and non-clustered versions, as Chapter 5 will explore further. Even though the input dataflow is very flexible and has several degrees of freedom, it should be mentioned that not every possibility is allowed and there are some boundaries. For instance, the step size cannot be too big in certain dimensions, since we would be skipping data for the convolution, nor to small since we would be duplicating data.

Figures 4.3 and 4.4 show a mapping example of the dimensions corresponding to its MAESTRO description. In this IS case, the number of filters (K) is spatially mapped across the PEs. In the first time step, every PE gets the same input values but different filters, and in the second time step the inputs remain stationary while each PE receives a new filter. Then, in the third time step, once all the filters have been iterated for the first piece of input, a new input is fetched by every PE and we start iterating the filters again.
In the WS case, however, the spatially mapped dimension is X (input column), and therefore now in the first time step all the PEs have the same filter but different pieces of the input. Since the first temporal dimension is Y (input row), at the next time steps the PEs will hold the same weight values but the input will change.

An interesting behavior that can be observed when mapping the dataflows into the PEs is the temporal folding. This phenomenon takes place when the amount of PEs is not enough to spatially parallelize an entire dimension. Taking Figure 4.3 as a reference, if the number of filters (K) is bigger than the number of PEs in the array, we will need at least two temporal iterations to fulfill the operation. In this case, the dataflow performance is highly impacted as well as the broadcasting opportunities. Therefore, this is an example of how we cannot predict the exact behavior of the accelerators unless we specify both the dataflow and hardware resources as well as the targeted neural net.

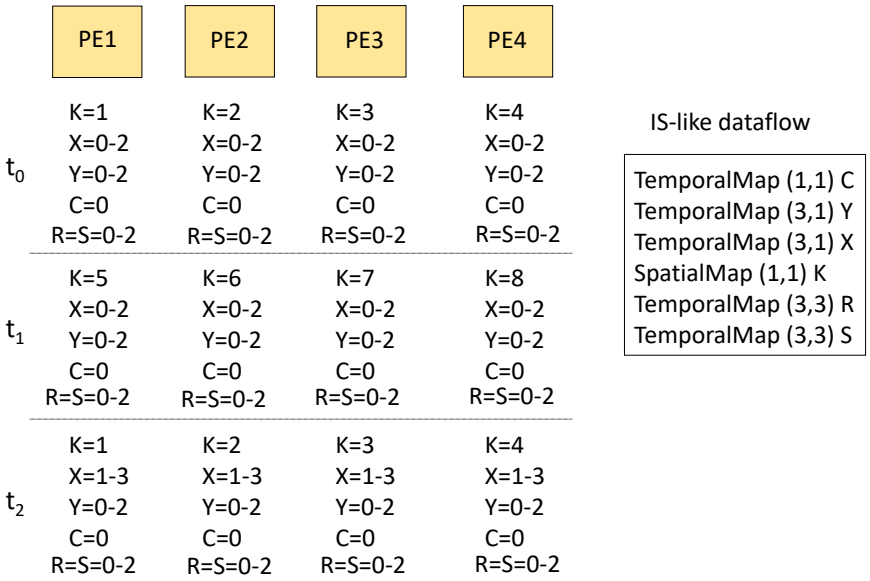


Figure 4.3: *IS-like dataflow mapping and MAESTRO description*

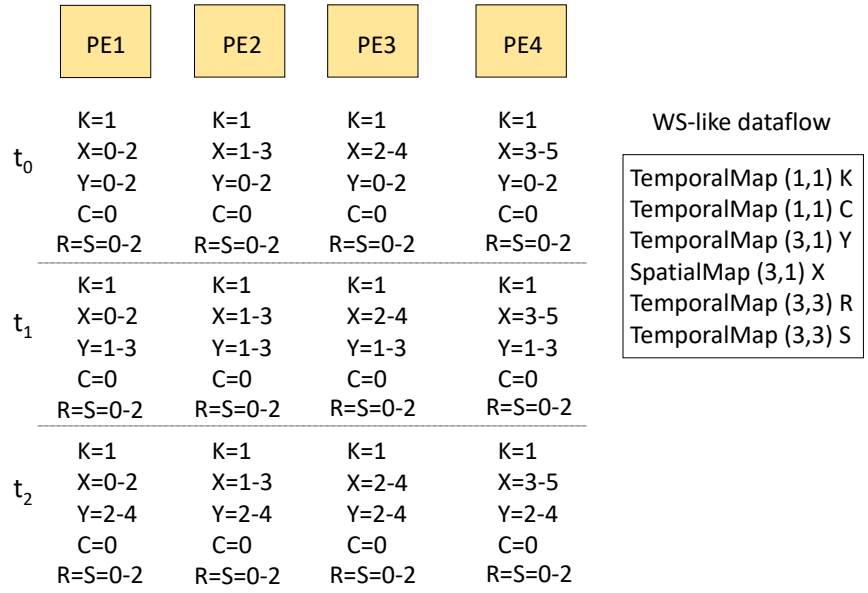


Figure 4.4: *WS-like dataflow mapping and MAESTRO description*

As it can be seen, the IS-like dataflow makes the PEs hold the same input values for more than one iteration (dimensions X,Y and C), and the WS-like dataflow makes the same thing with the filter dimensions K, R and S. These examples do not show clustered dataflows, and a further analysis on the data movement in those cases can be found in the Annex I chapter.

As the reader may have noticed, the dataflows comprehension and optimization is a deep and complex topic, and therefore a great part of this project has been spent trying to learn how to work with them

smoothly.

Unless otherwise specified, during this work we will talk about throughput in MACs/cycle units, being a representative measure of how many operations is performing the accelerator in a cycle. For instance, an accelerator suffering from starvation will have low throughput. Besides, the bandwidth of the network-on-chip will be defined in number of parallel elements of 8 bits that can be transmitted in a cycle.

For the NN models, during all the project the VGG16 [3], a convolutional neural network designed for classification, has been used for the evaluations and performance checks. The most used layers have been CONV1, CONV2, CONV4, CONV8 and CONV11, because they represent a set of variable sizes and shapes, as Figure 4.5 shows. As can be seen, the first layers do have big input dimensions while the late ones have bigger filter dimensions.

```
CONV1:   K: 64,  C: 3,  R: 3,  S: 3,  Y: 224,  X: 224
CONV2:   K: 64,  C: 64,  R: 3,  S: 3,  Y: 224,  X: 224
CONV4:   K: 128,  C: 128,  R: 3,  S: 3,  Y: 112,  X: 112
CONV8:   K: 512,  C: 256,  R: 3,  S: 3,  Y: 28,  X: 28
CONV11:  K: 512,  C: 512,  R: 3,  S: 3,  Y: 14,  X: 14
```

Figure 4.5: *A set of VGG16 layer dimensions*

4.2 Transceiver models

In order to evaluate the two technologies together - WNoC and DNN accelerators - a model of the wireless on-chip transceivers is needed. This will allow the comparison of datarate and bandwidth, as well as the power and area consumption.

To do so, real publications of transceivers at this scale have been studied to create a behavioral model from which to extrapolate and estimate the transceivers performance and overheads.
Another approach that could have been followed is to create a model out of the theory of transceivers design. However, when it comes to analog designs, the assumptions are not as accurate as we would need to create a model, since there are several degrees of freedom that the architect chooses, and every design is as unique as itself. The process of designing a wireless transceiver is not iterative, and it is difficult to built upon previous designs. Instead, most of the designs start from scratch, and if we were to obtain a theoretical model of the wireless transceivers, multiple circuitry considerations would not be met or could be wrongly assumed. Therefore, the analog devices are not trivial to model and some considerations must be taken into account.

For example, it can seem logical to assume a linear increase of the total power consumption with

the datarate. However, even though the transmitter power may be close to linear due to the amplifier consumption, there is a fix power consumption that may or may not follow that trend. In these devices, the linear approach of the power increasing with the datarate makes more sense for long range and high datarates transceivers, because the RF power dominates. However, at low datarates and short ranges, the electronics power dominate, as has been argumented in [58] for the Bluetooth case. Moreover, the same publication argues:

> "For higher datarates [than few Mega-bits per second], the assumption that the local oscillator, the receiver and the transmitter have constant power consumption no longer hold. In fact, the power consumption of these circuit blocks will eventually increase with data rate."

Yet another approach could still be followed. Instead of creating a model from published results, we could try to find the requirements of our transceivers for the DNN accelerator and then locate the best existing transceiver that fulfills those requirements from our pool of sample designs, that is the closest one to our needs. Nevertheless, even though this would facilitate the choice of the best state-of-the-art transceivers for the design and it would be very realistic, it would not generalize easily for DNN accelerators, and each design would require such repetitive analysis. Consequently, the approach of obtaining the model from publications has been followed.
In any case, this model gives us an approximation of the transceivers consumption to provide certain datarate. However, an implementation requires a specific design of the emitter and receiver, which could be facilitated by looking at the closest publication points once the target datarate is found.

After analyzing some of the state-of-the-art transceiver features [59–85], a fitting model has been obtained. The model has been split into area and power representations.

4.2.1 Area model

We are interested in modelling the area consumption of the transmitter and receiver to compare it with the wired interconnects as well as to compute the overheads of WNoCs.
After analyzing the published transceivers, optimistic and pessimistic trends have been computed. For the pessimistic trends, only the benchmark of on-chip transceivers has been taken into account, while for the optimistic trend, both on-chip and long range transceivers have been considered. Even though a long-range design process can be slightly different, because it may for instance seek for different modulations, the extrapolation is fair for comparison.
In addition, for the optimistic fittings a Pareto front [86] has been computed from the results. By finding this Pareto set, we make sure that our optimistic model takes into account the best designs, which is something the approach of the pool of transceivers would have granted us.

Moreover, a weighting vector, defined as the maturity factor, has been computed and applied over the fittings [44]. For each transceiver, we can find this factor as $MF = R_b(bps)/f_c(Hz)$, being R_b the datarate of the designed transceiver and f_c its carrier frequency. In a sense, this parameter gives us an idea of how much developed that transceiver is, and it is closely related to the spectrum efficiency. Therefore, it makes sense to give more importance to those state-of-the-art transceivers.
Figure 4.6 shows the obtained model.

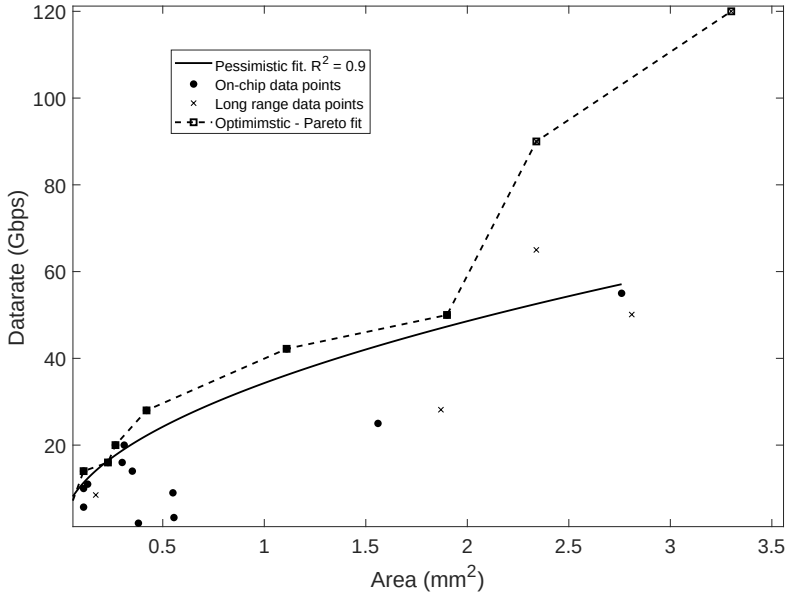


Figure 4.6: *Area model [59–85]*

Higher frequencies tend to facilitate an easier allocation of the bandwidth, thus easing the designs with wide datarates. However, even though a certain transceiver works at high frequencies, it will not grant us high datarates until that frequency range is developed, or matured. In this scenario, not only the technological research has an important impact, but also regulations and laws enter into the equation.

For instance, as Figure 4.7 shows, the 60 GHz frequency carrier is one of the most developed ones, due to the recent opening of that spectrum for license free operation [87], which increased the demand of the millimeter wave in several applications. Moreover, this carrier has been proven to achieve the usually required datarates. Around 200 GHz, new developments are starting to arise, which have potential improvements over the current used bands. This is why we chose to include the maturity factor in the model.

It should be mentioned that some of the transceiver parameters have been inferred, in most cases for fairness. For instance, the areas provided in some papers included pads or extra layouts, but only the active core area has been used to compute the models.

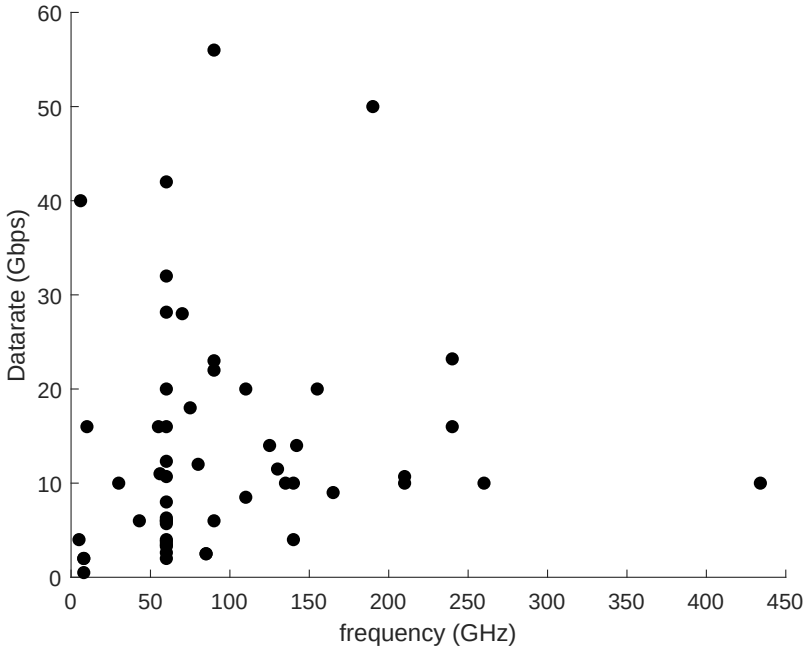


Figure 4.7: *Frequencies maturity*

4.2.2 Power model

To measure the WNoC overheads, the power model to relate datarates with consumption is needed. In this model we have also computed optimistic and pessimistic trends like we have done in the area model, but certain extra considerations have been taken into account.

First of all, a normalization factor has been applied for fairness. This is because when comparing different transceivers, some of them may have been designed to reach different ranges or to have different Bit Error Ratios (BER). These changes in the design choice affect the power consumption, since the components, mainly the amplifiers, will spend more resources to work on longer communication distances or fewer errors, by radiating more power. Therefore, the power has been standardised in an attempt to compare in a more fairly way.
Taking a look into the power budget link Equation (4.1) we see how the power depends on the distance.

$$P_r = \frac{P_t \cdot G_t \cdot G_r}{L(d^\alpha)} \tag{4.1}$$

where P_r is the received power, P_t is the transmitter power, G_t and G_r are the transmitter and receiver gains. α represents the path loss exponent, d is the distance and $L(d^\alpha)$ is the attenuation or path loss.

Specifically, the power has been normalized as $\frac{1}{\sqrt{distance}}$, as suggested in [88], even though valid α parameters can be in the range of [0.5 - 1] due to the placement in a closed package. A deeper analysis in the package channel can be found in [40].

Similarly, looking at the BER[1] Equation (4.2), we see how it also depends on the power.

$$BER^{BPSK/QPSK} = Q\left(\sqrt{\frac{2 \cdot E_b}{N_0}}\right) = Q\left(\sqrt{\frac{2 \cdot P_r/R_b}{N_0}}\right) \tag{4.2}$$

where $Q(.)$ represents the tail function of the standard normal distribution, E_b is the bit energy, R_b is the datarate and N_0 is the noise power spectral density.
Therefore, every transceiver has been also compensated for BER differences.

As a result, Figure 4.8 shows the power model with the normalization factor applied.

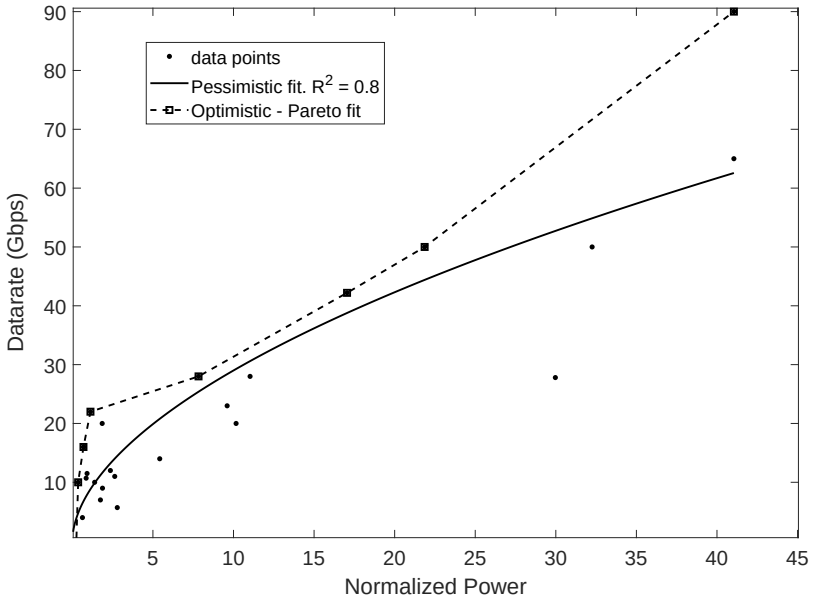


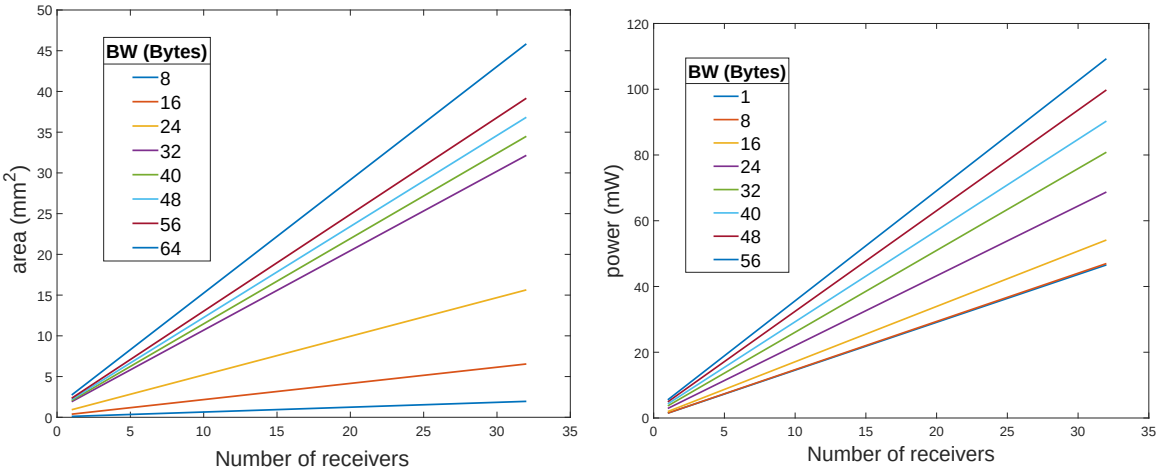
Figure 4.8: *Power model [59–85]*

[1] Assuming AWGN noise, which is the worst-case scenario [89].

After having obtained the transceiver models, an analysis has been carried out to find the distribution load of the power and area of the different implementations into the transmitter and the receiver.
In a general manycore environment, where all the nodes send and receive data, the allocation of the complexity in the trasmitter or the receiver is not so critital. However, since we want our implementation to take advantage of the broadcast opportunities in the distriution step of the layer computation, our design will likely have few transmitters and multiple receivers. After comparing the sample points, an average of 50.6% of the area, and a 46.6% of the power were consumed in the receiver and the rest in the transmitter.

Nevertheless, this is a design choice because the complexity of the transceivers can be given over the transmitter or the receiver as desired. For instance, the transmitter amplifier can have more gain at the expense of a reduced consumption in the receiver to maintain a safe link budget, as Equation 4.1 indicates. As an optimistic reference, a 35% load for both power and area consumption has been assumed in the wireless receiver.

We can now observe the linear tendency we talked about earlier when increasing the number of cores or receivers, as Figure 4.9 shows.



(a) *Area* (b) *Power*

Figure 4.9: *Optimistic wireless network consumption as a function of the receivers*

Finally, even though it has not been used in the definitive implementation, a technology gap analysis has been carried out to downscale the technology. Since one can not assume the models to follow a linear trend, this analysis has been conservatively done using a sub-linear approach.
In this sense, from 65nm technology to 28nm technology a 2X factor reduction has been applied to obtain the power and area consumption of those transceiver implementations.

Now, with Figures 4.6 and 4.8 we already have the wireless transceivers well characterized. This models will be used later on to compute the overheads of augmenting a DNN accelerator with WNoCs.

5. Design Space Exploration

In this chapter, an exhaustive understanding and search of the different design parameters in DNN accelerators will be carried out. The mainly tuned variables can be found to be the bandwidth, the dataflows, and the number of processing elements. In Section 5.1, a first on-chip design will be evaluated. In Section 5.2, an off-chip WNoC will be explored and in Section 5.3 a hybrid design will be discussed.

5.1 Wireless On-chip

The main objectives that have been taken into consideration have been to allow the accelerator to broadcast data easily and in a flexible manner, using dataflows that require low bandwidth, relying in an adequate number of PEs (that is, not too many since they would be underutilized but not too few so that we can not paralellize enough the data dimensions) and power and area savings.

The first approach to augment a DNN accelerator with WNoC support has been the inclusion of wireless transceivers at PE-level, as Figure 5.1 shows. In this case, the on-chip shared memory should have the transmitter and each PE would need a receiver.

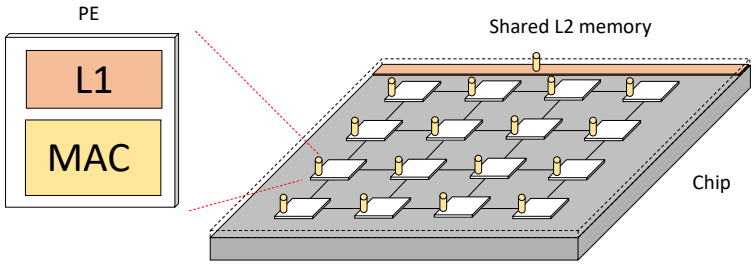
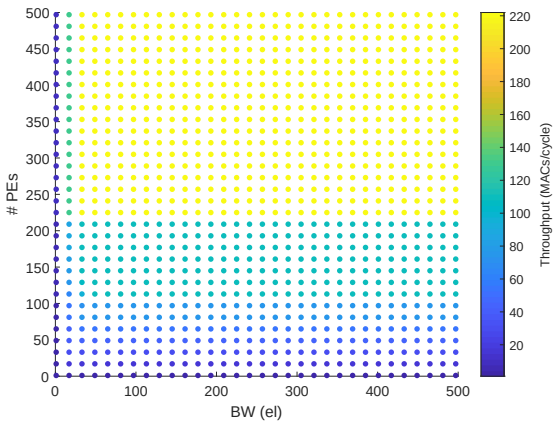


Figure 5.1: *Wireless On-chip implementation*

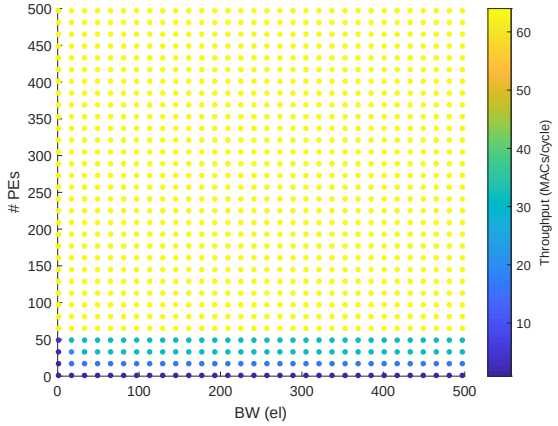
In the on-chip case, as Figure 3.6 shows, different dataflows will have different behavior at different

layers, and therefore we need that flexibility in the topology. On-chip wireless networks allow this at an area and power expense.

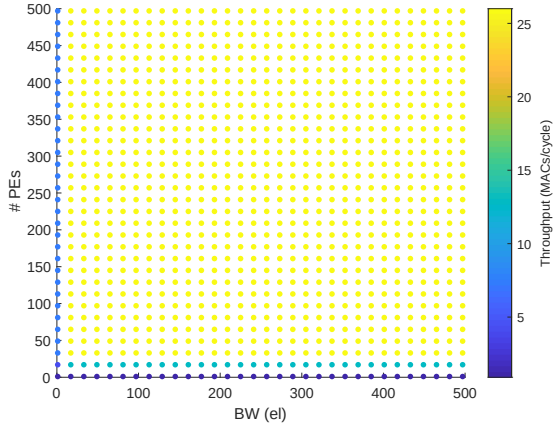
To better understand our possibilities, the throughput has been mapped at different layers for different dataflows, iterating the number of PEs and the bandwidth of the NoC. Figure 5.2 shows a small but representative sample of the obtained results. For each different set of number of PEs and bandwidth points we show the throughput at a specific layer and under a certain dataflow. Since we are at PE level, these are all non-clustered dataflows. In the figure, darker colors indicate poor throughputs, whereas brighter colors exhibit points where the throughput is better. The most desirable points in these plots are the brighter bottom-left ones, since they give us the best throughput at the minimum bandwidth and number of PEs required.



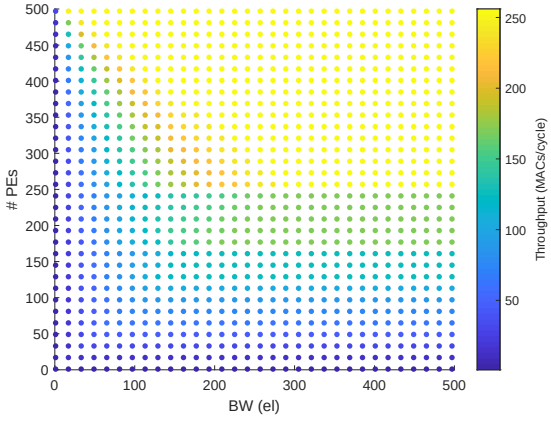
(a) Layer: CONV1. Dataflow: WS-like



(b) Layer: CONV1. Dataflow: IS-like



(c) Layer: CONV8. Dataflow: WS-like



(d) Layer: CONV8. Dataflow: IS-like

Figure 5.2: On-chip design space iterations. Note that the scale of each figure is different.

An important limitation of the DNN accelerators is that the throughput is a function of the PEs. If we have N PEs, our throughput upper bound will be N MACs/cycle, assuming a single MAC unit

per PE. This gives us an insight of how well a dataflow is performing by looking at how close its throughput is from that limit. Taking a look at Figures 5.2a and 5.2d, we can see how as expected a simple WS-like dataflow reaches good throughputs at early layers, as well as IS-like mappings do at late layers. However, even though the color scale may be tricky, Figures 5.2b and 5.2c show bad performance by saturating far from the theoretical limit, around 60 and 25 MACs/cycle respectively.

As it can be seen, even when increasing the PEs, the throughput does not keep increasing after a certain number of MACs/cycle, mainly due to dataflow parallelization limitations. One way to see this is by taking a look at the spatial reuse. Since we have a dimension to spatially map, once we can deliver all the values of that dimension to different PEs (that is, we can totally parallelize that dimension), increasing the number of PEs above the size of that dimension is useless. A similar thing happens with the bandwidth: in most cases it does not help to increase it sharply without increasing the PEs. Depending on the dataflow, the number of PEs can be greater than the bandwidth if there is reuse. However, depending on the tiling size, it is also possible to have more bandwidth than the number of PEs, for instance to transmit weights and inputs at the same time. In general, though, having much more PEs than bandwidth capability will lead to starvation and having much more bandwidth than PEs will cause lack of parallelization.

Another key point we can observe is that at certain points, the throughput suddenly soars, taking a huge step. These points indicate a strong parallelization change of the spatially mapped dimension, and thus usually happen at a number of processing elements power of two.

In an on-chip environment, with a reduced number of PEs, the traditional NoCs are well optimized and can fulfill most of the network requirements the accelerator needs. If we take a look at the energy efficiency levels, defined as how many Joules per sent bit are being spent, the wireless NoC outperforms the traditional NoC only for broadcast traffic and when the number of receivers is high enough. That is because when WNoC broadcasts, the transmitter can save power during some time slots, and when the number of cores is high enough, the WNoC power scales better. With our transceiver models we could check that, as Figure 2.11 shows, wireless energy efficiency is around 2.8 pJ/bit[1]. However, when unicast traffic has to be mapped, traditional NoCs perform better at this scale.

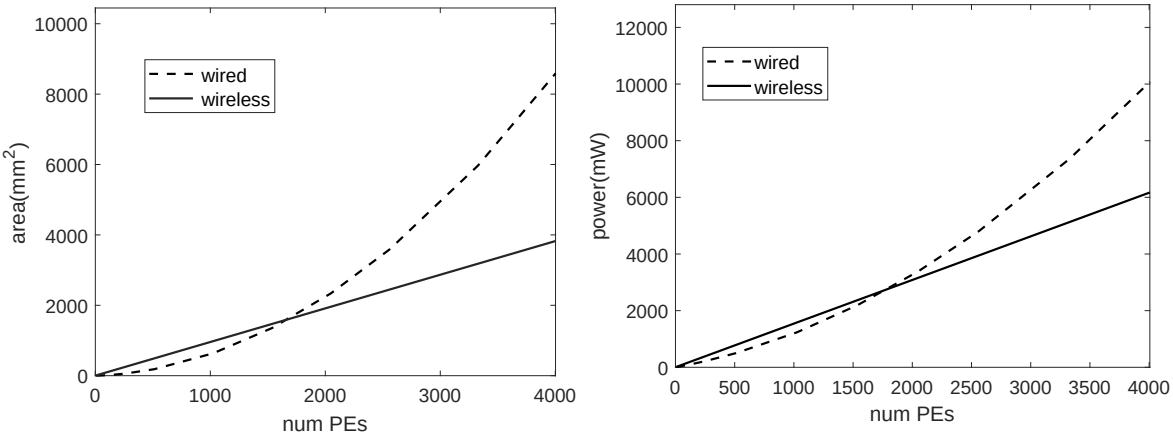
As already mentioned, Figure 5.2 shows how the performance saturates at a relatively small number of PEs, due to limited parallelization, which depends on the dataflow and the mapped layer. However, as we have argued in Chapter 2 and looking at Figure 5.3, the benefits of WNoCs are better exploited when the number of cores increases. Therefore, we are in a deadlock situation because WNoC would be useful with a number of PEs that is unnecessary for these dataflows. By using simple dataflows, increasing the number of PEs does not improve the throughput, but without increasing the number of

[1] As an example, a 100 Gbps receiver and transmitter have been taken, which consume around 282.8 mW for a 1cm link distance at a BER equals to 10^{-5}.

receivers the WNoC consumes more than traditional NoCs.

In other words, when comparing a bus NoC and wireless transcievers to match the worst-case unicast demands, WNoCs scalate much better than the NoC at a high number of cores, but the surpassing point at which it is profitable is way above the usual required number of PEs inside a chip in accelerators, both in area and power senses.

In an on-chip environment, the bandwidth is usually not a limitation and the wireless interconnects do not improve it, but the flexibility is. Therefore, the on-chip network could benefit from the broadcast and flexibility of WNoC, but it may be unrealistic since unicast traffic is not negligible at onchip level and the datarate requirement increases, being the onchip NoCs more capable to provide it at these amount of PEs for now.



(a) Area

(b) Power

Figure 5.3: *On-chip NoCs comparison. BW = 64*

By implementing a realistic chiplet, with on-chip wired and wireless interconnects and same bandwidth, the area of the latter is between two and ten times greater than the former one, by using the models previously mentioned.

In sum, in this case we did not find the WNoC support to be of a great advantage at PE level.
Despite that, we said earlier that DNN accelerator PEs tend to be clustered in practice. This other level will be studied in the next section.

5.2 Wireless Off-chip

In the previous section we compared Wireless NoCs to traditional NoCs at PE level, where the wired topologies tend to perform better because of limited number of PEs and the optimized electric NoCs.

However, as Figure 5.4 shows in a simplified way, full DNN accelerator systems tend to be formed by clusters of PEs, creating a new level that requires interconnections as well. Therefore, we can now understand every cluster as a small chiplet that is in itself a small accelerator.
In this case, the dataflow search exploration becomes even more flurry. When we had no clusters, it was clear that, for instance a WS-like dataflow, would work fine with layers having large inputs. In that case we could statically set different weights at each PE and we could broadcast, or spatially reuse, the large inputs. Nevertheless, when adding clusters we can see in some cases WS or IS behaviors depending on the level we are looking at. This makes it non-intuitive and we must carefully analyze the possibilities. Even though for this analysis a single level of clustering has been considered, the possibilities for the dataflows go beyond this and allow several levels of clustering.

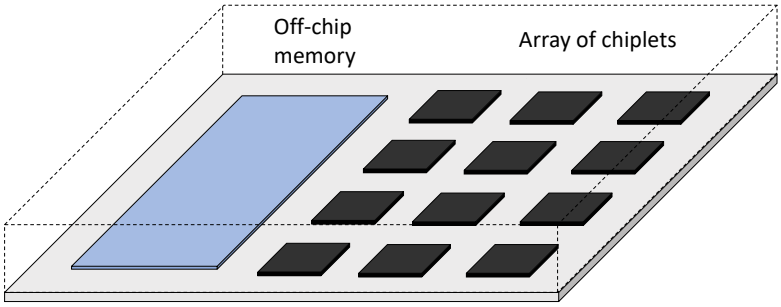


Figure 5.4: *Simplified schematic of a DNN accelerator*

One of the problems we had at PE level is that we suffered from limited parallelization due to the dataflow, and thus placing more PEs was not useful to increase the throughput. In this off-chip scenario, however, the dataflow can include clustering, which opens new possibilities. The direct impact on the dataflow is that we can now spatially map different dimensions at each level, increasing the parallelization.

For instance, as Figure 5.5 shows, we mapped the same VGG16 neural network into 16 clusters of only 16 PEs each. However, we achieve throughputs really close to 256 MACs/cycle in every layer, which is the theoretical limit. For this representation we chose the best dataflows performing in each layer, which require different parallelization strategies. A full description of the dataflows is not included here for the sake of brevity, but as an intuition the best dataflows map spatially over the different levels the dimension whose size is bigger, because they will allow better parallelization opportunities.

Therefore, we can now increase the total number of PEs to keep increasing the throughput, and it will saturate at a higher throughputs.

Shifting to the hardware support, while on-chip networks are well suited for the PE level, at off-chip level the need to travel long distances is making traditional NoCs to lag behind [90, 91].

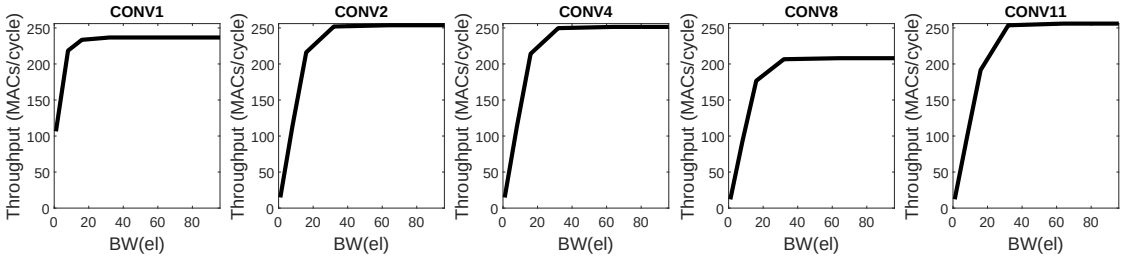


Figure 5.5: *Throughput analysis with clustered dataflow*

In this scenarios, off-chip NoCs still have to face scalability challenges [92]. Most of these networks in accelerators implement neighbor to neighbor or point to point communication links because other options are costly and prohibitive. This led most of the design approaches to develop towards that restricted scheme [93].

A recent approach for wired NoCs implementation is to rely on interposer networks, by placing an active NoC chiplet on it, as Figure 5.6 shows. This technology allows the so called 2.5D IC designs, by interconnecting independent chiplets using a passive interposer fabric beneath them [94]. This way, the connections do not need to go all the way down to the die and to be mapped across it, which is power consuming, and go up again. Instead, the interposer allows fast communication between chiplets and only those connections leaving the SoC need to be mapped down. Moreover, each chiplet can be manufactured separately and using different technologies [95].

Notwithstanding, its limited space availability does not allow for broadcast or long range communications. In this sense, the active NoC chip, typically a mesh, routes the traffic through thin links to the chiplets, which turns in limited bandwidth besides being a bottleneck for scalability and broadcast, due to the cost of placing the bumps [96, 97]. We have to take into account that both the chiplets and the interposer have limited room for pins, limiting the bandwidth [28, 98], also to access the memory [99]. This is because the pitch of the I/O pins in IC is not scaling as well as on-chip, and it limits the the feasible topologies. In this sense, most of the accelerators work with 64 Gbps links at 1 GHz frequency clock, which means an element bandwidth of 8 bytes [100].

As can be read in [28]:

> "By moving on-chip, the I/O bottlenecks that faced prior multi-chassis interconnection networks (off-chip) are alleviated substantially: the abundant on-chip wiring supplies bandwidth that is orders of magnitude higher than off-chip I/Os while obviating the inherent delay overheads associated with off-chip I/O transmission. [...] Due to the abundance of on-chip wiring resources, channels tend to be wider in on-chip networks. In off-chip networks, channel widths are limited by pin bandwidth."

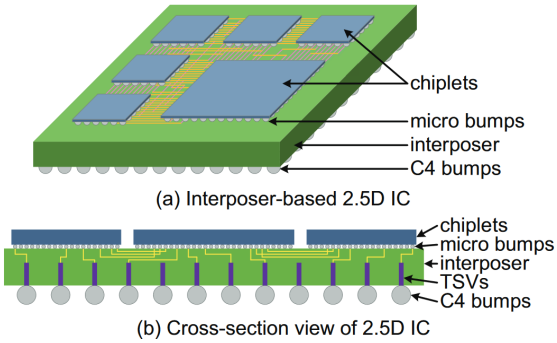


Figure 5.6: *Interposer-based NoC [96]*

In this scenario, WNoC could improve these limitations, since it enables both long range communications within package and can provide higher bandwidth than interposer-based NoCs, even between detached chips, while supporting multicasting natively [101].
Moreover, by implementing off-chip communications, the overheads of the transmitter and the receiver are not so heavy as at the PE level. Even though we may still have a small number of receivers, which will cause some overheads, the increased bandwidth and the long distance capabilities give WNoCs an advantage over traditional off-chip networks, as Figure 2.11 shows. The off-chip energy efficiency comparison to wired NoC is expected to be improved with respect to the on-chip case, due to the longer links, which makes wired power dissipation to supralinearly increase [51, 56, 92]. For instance, wired off-chip energy efficiency is estimated as 5-10 pJ/bit in [99].

It should be mentioned that off-chip WNoCs and interposer networks for inter-chip communications do not exclude each other. That is, including wireless links is agnostic to the underlying technology, either a substrate or an interposer. Although not studied here, chip-to-chip communications could be facilitated by these two technologies working together to improve spatio-temporal reuse.

To carry on with the design, we chose a target throughput of 16384 MACs/cycle, which can be theoretically achieved by using 16384 PEs. To do so, we have now different cluster size options. For instance, we could architect an accelerator with 1024 chiplets of 16 PEs in each, 512 chiplets of 32 PEs in each, and so on. An extreme case would be a single huge 16384 PEs systolic array-like chiplet. The optimal distribution highly depends on the dataflow used and the target neural network. In this case, a significant amount of dataflows with clustering instructions have been iterated in MAESTRO to carry out a small dataflow space exploration.

The variants in the tested dataflows do not only include WS-like and IS-like behaviors, but also rolled and unrolled strategies, changes in the order of the mapped dimensions and tiling sizes, and so on. The set of simulated dataflows has been selected to try to have a wide range of options, including extreme cases.

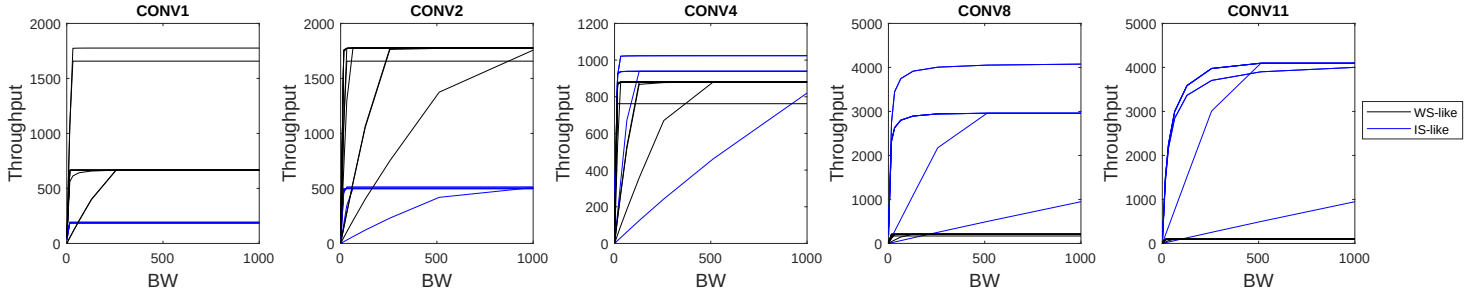
However, the iteration of the dataflows can not be random. For instance, we have to avoid illegal dataflows, as explained earlier in Chapter 4, or redundant ones. An example of two redundant dataflows could be, for example, a single order shift between dimensions X and Y, since they relate to the same data class (inputs) and their size in the target neural network is always the same, as Figure 4.5 shows. In this case by redundant we mean that the hardware metrics, such as throughput or runtime, will not change, even if the dataflows are actually different and the mappings differ.

In Figure 5.7, a comparison between the dataflows in each layer can be found. This study shows the difference in performance of different dataflows in different layers, when changing its off-chip bandwidth. The blue traces show the computational throughput of a set of IS-like dataflows while black traces show the computational throughput of WS-like dataflows. Different cluster sizes are shown, which lead to different throughput for every dataflow. In these cases, cluster sizes of 8, 64, 256 and 1024 number of PEs per chiplet have been tested. As can be seen, the impact in performance is huge and some choices are more valid than others. In some cases, when the cluster size is too big the dataflow choice led to PE underutilization, while in other cases there is lack of computing parallelizable resources.

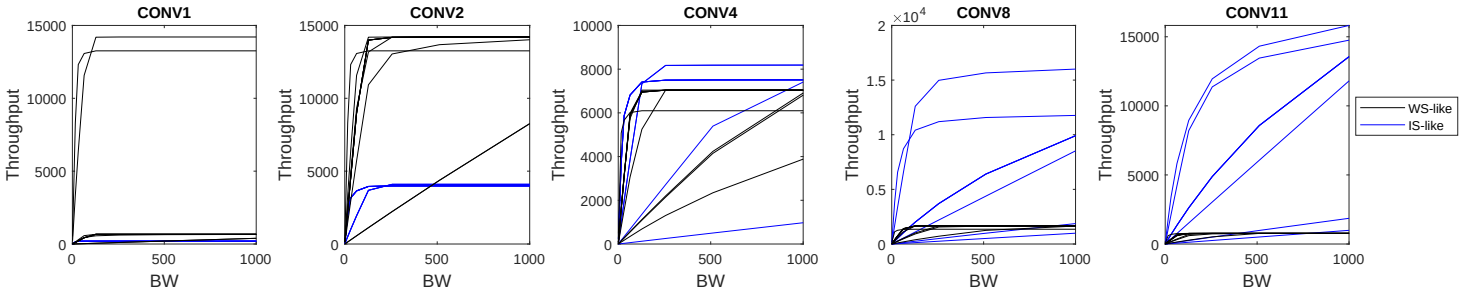
We can also observe that some dataflows perform really poorly in all layers. This may be for several reasons. For instance, the spatially mapped dimension might be substantially smaller than the number of PEs, and we could be underutilizing them. Another reason could be that we are temporally mapping a coupled dimension such as the input channels, leading to poor reuse.

An important point that can be seen is that some dataflows perform better than others depending on the bandwidth. That is, a certain dataflow can be the best choice in low bandwidth situations, but when we have the opportunity to deliver more data, that dataflow is not the best one anymore. Thus, there are certain interesting surpassing points in which not only the throughput is boosted due to the bandwidth improvement, but also due to the enabled use of more profitable dataflows.
If a dataflow needs less bandwidth, it may be an indicative that it is broadcasting more data. This is of an utmost importance in our case, since interposer-based NoCs tend to be limited to low bandwidth for long range links, whereas WNoCs bandwidth is independent of the distance at this level. Current WNoC technologies can provide up to 64 bytes bandwidth at 200 MHz clock, while most of the wired off-chip NoCs provide a bandwidth of 8 bytes. Moreover, it should be noticed that these graphs also show the potential of the WNoC in DNN accelerators when bandwidth capabilities would be improved.

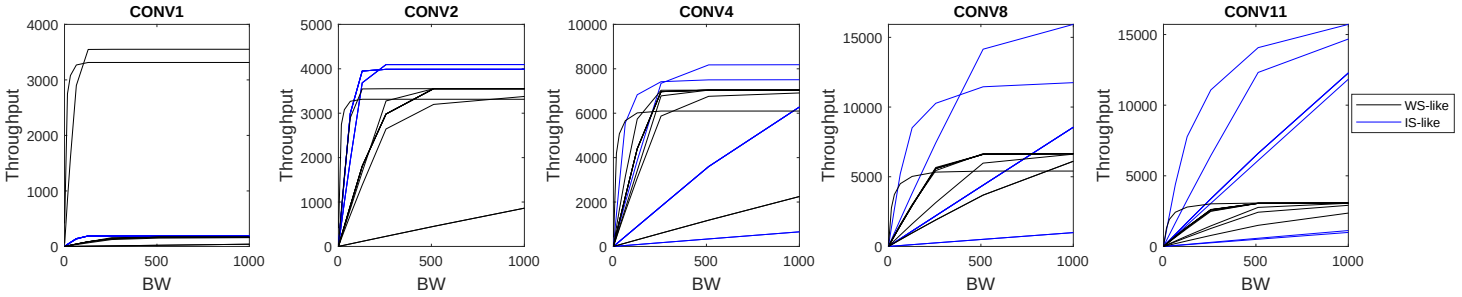
However, these advantages come at certain drawbacks. Firstly, there will be overhead cost in terms of area and power. Moreover, the interposer NoC allowed neighbor to neighbor communications, which in certain situations such as spatio-temporal reuse could be very useful. This is not supported by our implementation of WNoC because we are not including transmitters at chiplets. In addition, certain dataflows in which unicast traffic prevails are not adequate at all for this implementation.



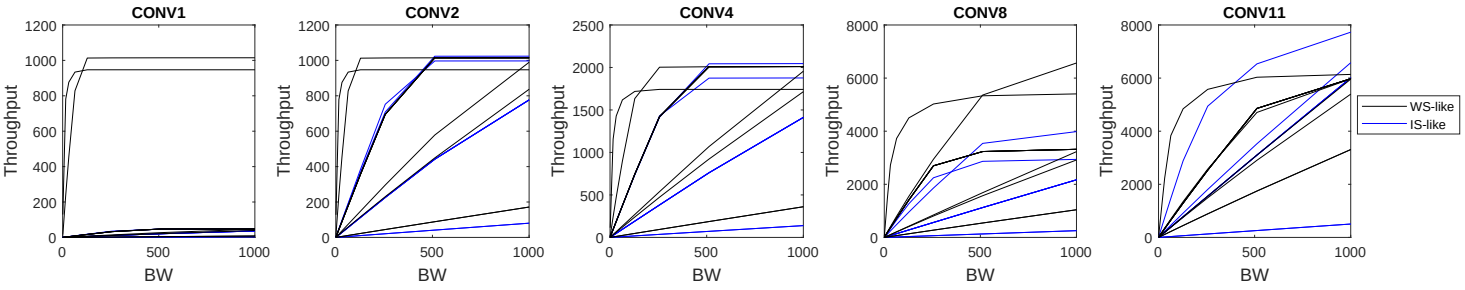
(a) Cluster size 8



(b) Cluster size 64



(c) Cluster size 256



(d) Cluster size 1024

Figure 5.7: *Throughput analysis for different cluster sizes and dataflows*

5.3 Hybrid accelerator

Even though it has not been developed or analyzed thoroughly, the idea of an hybrid accelerator has been part of the design exploration process, and will be summarized here.

The main benefit of WNoCs, as compared to traditional NoCs, is the possibility to deliver fast broadcast data to every point in the package, but it suffers in situations where unicast traffic prevails. Therefore, it seems logical to combine both wired and wireless channels to transmit certain data through the WNoC and other data through the traditional NoC. For instance, in a WS-like dataflow that would allow input values broadcast, we could use the WNoC to send that data to the PEs but still use the traditional NoCs to transmit the filter weights. In another case, however, we could broadcast weights and use the wired NoC to unicast inputs. This would still allow some flexibility in terms of dataflow, since we could shift which data is sent to each channel. In the end, including WNoC in DNN accelerators is about taking advantage of the transceivers benefits, and we could cover every case with the technology that best fits that kind of traffic. In addition, we could split the bandwidth requirements between those both channels, and the wireless datarate would not need to be so high.

In a sense, the wireless network could also be used to simplify the wired network. For instance, in crossbar or tree-based designs, if we remove the need of broadcast support from that wired network the switches and nodes can be simplified.
Moreover, the wireless NoC could be also used to configure the wired NoC, if it was flexible, and it could boost the adaptability of it. For example, routers that change traffic patters in the wired NoC could be listening to the wireless channel to be set in certain configurations by making use of lookup tables. Current flexible NoCs need several cycles to be set properly, whereas by using a wireless broadcast it could potentially be done in a single cycle.

Although this is not the main focus of this work, an implementation can be found in [102]. However, in this case only a static dataflow and a single WNoC transceiver have been evaluated.

6. A wireless-based DNN accelerator

In this section, a specific accelerator design will be discussed and evaluated.

6.1 Design and evaluation

Taking into account the design space explored in the previous section, a specific DNN accelerator has been theoretically designed by means of augmenting it with wireless interconnects.

By taking a look at the analysis carried out in Figure 5.7, a cluster size of 64 PEs per chiplet has been chosen. As can be also observed in Figure 6.1, this cluster size provides a better throughput in every layer, naturally by using different dataflows. This is because the dimension K of the targeted neural network, which refers to the number of weights and it is not coupled with the inputs, is easy to parallelize, as the example in Figure 6.3 will later show. Since this dimension size is always multiple of 64, as can be seen in Figure 4.5, foldings are well parallelized. A similar thing happens with the channel dimension, and if we had unlimited bandwidth we could achieve throughputs really close to the theoretical limit. Since we targeted a number of 16384 PEs for our design, the die will contain a total of 256 chiplets.

The specific DNN accelerator design will consist of a High Bandwidth Memory (HBM) feeding a global SRAM buffer at off-chip level. This buffer will be augmented with a wireless transmitter working at a bandwidth of 64 byte elements per cycle, which is equivalent to 102.4 Gbps at a clock frequency of 200 MHz. The antenna will be a small Through-Silicon Vias (TSV) monopole placed on top of it, in an attempt to save area while having omnidirectional support for the broadcast.

In the chiplets side, every chiplet will be augmented with a wireless receiver and antenna. The inside implementation of each chiplet is based on 64 PEs, but the NoC itself is not specified and it is going to be a final design choice. For our evaluations, we used a bus-based NoC, since it is broadcast friendly and MAESTRO supports it, and in the figures we show a systolic array-like NoC.

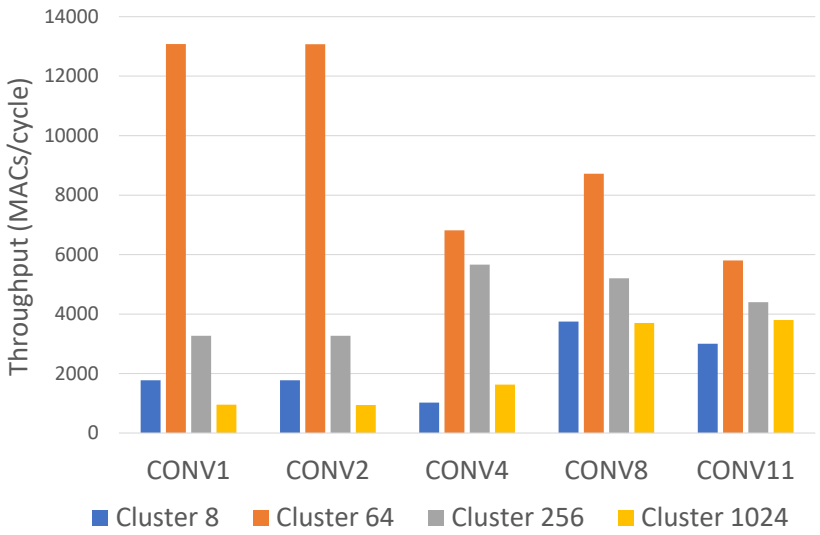


Figure 6.1: *Best dataflows throughput comparison at BW=64*

Even though a specific design is shown and evaluated, this architecture pretends to be generalizable, being it more powerful than a single implementation given that the on-chip NoC, and other parameters such as the dataflows or the WNoC bandwidth should be set as desired.

In Figure 6.2 a sketch of the system is shown.

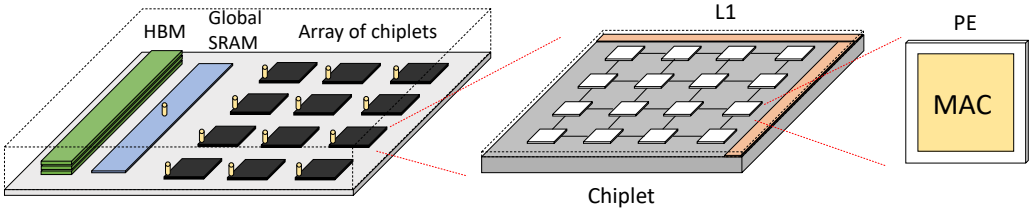


Figure 6.2: *DNN accelerator off-chip implementation*

As it has been one of the main concerns during the whole work, this design allows different behaviors depending on the dataflow at each level.

Figures 6.3 and 6.4 show a mapping example of a WS-like and a IS-like dataflow into chiplets at the accelerator. We make use of the flexibility that WNoC provide to facilitate the change in behavior of the data movements. When an IS-like dataflow allows weight broadcasting, the WNoC can send the data at low latency and in few cycles, achieving good energy efficiencies. Each PE in the figure is working in a different piece of input, and therefore a different output value. Similarly, when the system is computing a layer which needs a WS-like dataflow to be at its best performance, it can adapt to it and change its dataflow, now broadcasting the inputs. Each PE will then be working in a different filter, and therefore a different output channel.

Depending on the dataflow, the spatial reuse can be enabled due to the broadcast support within chiplets, and the temporal reuse is possible since we have buffers at chiplet level. Even though off-chip WNoCs as we have implemented them would not allow spatio-temporal reuse between chiplets, an inner NoC could allow that between PEs.

Since we have a 64 bytes bandwidth WNoC, each cycle can be partitioned in 64 time slots. This way, in the worst unicast case we could send 64 different bytes to different chiplets in a cycle. The idea is that broadcastable data will be sent directly in a single slot, whereas unicast data will be serialized to be fitted into one or more cycles in time slots. Multicast traffic, then, will be a point in between. This way, receivers will only consume power when they are actively listening in their time slot within a cycle. To make it work, a serializer and deserializer, which have minimal area and power consumption such as [103], should be included in the design.

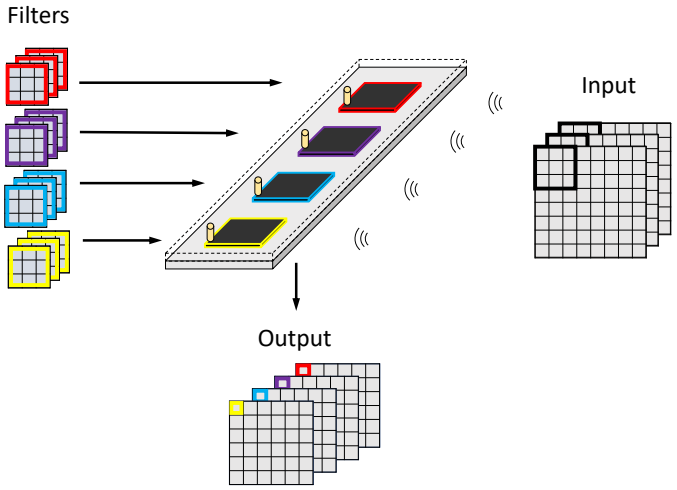


Figure 6.3: *WS mapping example*

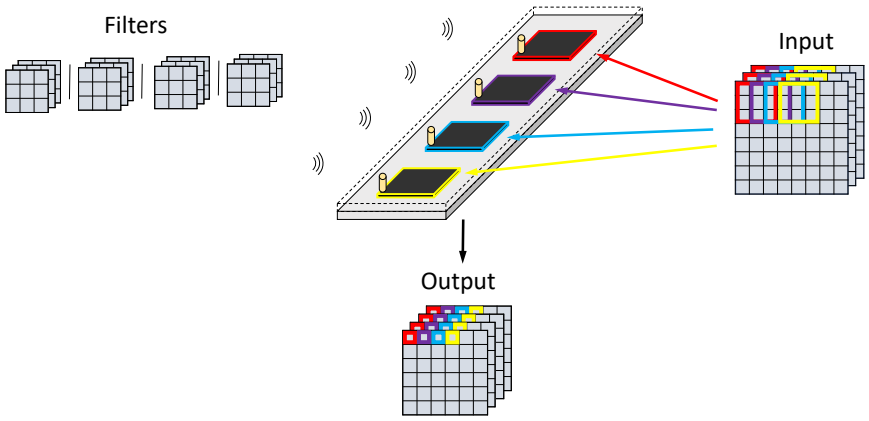


Figure 6.4: *IS mapping example*

Again, for the sake of simplicity, the figures only show a limited number of filters and inputs, and specific dataflows, but the idea can be extrapolated to any number of inputs and filters, and most of the dataflows and sizes.
Moreover, the off-chip communication is improved in terms of bandwidth. When unicasting - not an ideal scenario for WNoC - it still enables using a better BW than traditional NoCs, which as Figure 5.7 shows, leads to a better performance.

We then evaluated the potential of this architecture. We analyzed and iterated through different mapping strategies to find the best performing ones in the WNoC paradigm, that is to find the ones that can take the most out of the off-chip broadcasting support and off-chip high bandwidth, to define an adaptive dataflow. This dataflow is made of the best parts of each dataflow that showed better results in each layer in the analysis.

For comparison purposes, this architecture is contrasted with others that employ static dataflows and also with wired off-chip NoCs within the same dataflow. To do so, we simulate it in MAESTRO under the different dataflows and available bandwidth for every case, leading to the results shown in Figure 6.5.

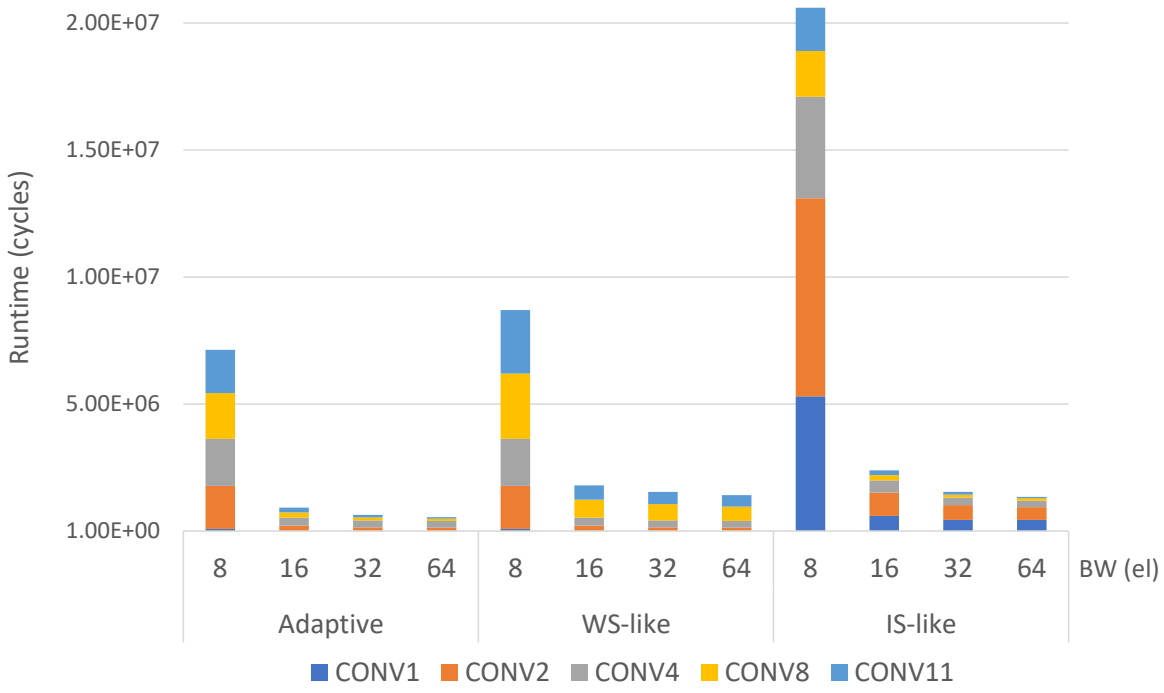


Figure 6.5: *Runtime evaluation*

Every column of a dataflow set corresponds to a different off-chip NoC bandwidth, where traditional NoCs tend to be of 8 elements[1] and WNoCs can even reach 64 element bandwidth, as we included in the implementation. Every column in the figure is indeed a stack of runtimes in which every color shows how much of a runtime each design has spent in each layer. As can be seen, the adaptive dataflow that WNoC facilitates will take as a runtime the sum of the lowest runtimes in each layer as compared as to fixed dataflows.

Overall, the runtime is reduced up to 13.4X, and an average of 9.9X, when compared to using a traditional NoC by assuming the same dataflow, due to improved bandwidth.
By taking advantage of the support of flexible dataflows against other static systems, we can see an average improvement of 2.2X assuming the same bandwidth.

However, it has still some limitations, especially in the unicast traffic case. Even though we iterated the dataflows to find the ones that broadcast the largest number of data, some unicast is always needed. In this case, the WNoC serializes the values and still broadcast them. Moreover, the receivers then need to know when a certain transmission is targeted at them or not. This control is not included here, and could be done by some adding some logic - like wired controller or wake-up radio - or by adding a protocol in the transmissions. The wired gather NoC, from chiplets to memory, has not been included either, since WNoC would not improve it. However, we could assume a 30% of overhead in the total consumption when including the wiring [96].

In order to quantify the area overhead due to the wireless transceivers, we simulated a 64 PE chiplet using MAESTRO, to compute values of a baseline chip. Each cluster, with non-blocking bandwidth, resulted to consume 6.9 mm^2 of area. Therefore, by augmenting them with the wireless receiver, it would represent a 12.16% of the total chiplet. If we take as a baseline the ShiDianNao [104] accelerator chip instead of our bus-based cluster, chiplets of 64 PEs consume only 4.86 mm^2 due to the reduced mesh-like NoC in their inside. This allows spatio-temporal neighbor reuse but no long distance or broadcast communications inside the chiplet. In this case, the optimistic overheads in the chiplets would be of 16.43%.
Finally, we choose the chiplet from the Eyeriss accelerator as a reference, as it has been widely used for comparison in literature [105–107]. After scaling it properly to match the same number of PEs, it consumes 4.65 mm^2, and therefore the wireless transceiver represents a 17.04% of the chiplet.
In the memory side, the global SRAM consumes 740 mm^2, meaning that when the transmitter is included it represents a 0.24% overhead. By taking a look at the pessimistic values, that is the conservative transceiver models and an equitable complexity distribution between receiver and transmitter, it would represent the 48.3% of the area at the receivers and 0.9% in the transmitter.

[1] Some rare examples reach 16 elements bandwidth.

The power overhead is trickier to compute, since it is not fixed and highly depends on the dataflow. If the transceivers were at full consumption, the maximum power overheads would be of 98.7 mW in the receivers and 183.3 mW in the transmitter[2]. By taking the pessimistic values, 177.4 mW would be added in the receivers and 208.2 mW at the transmitter. If we take the ShiDianNao accelerator chiplets, this overhead represents a 55% of the chiplet. Our Eyeriss chiplet reference consumes 89.9 mW, meaning that including the wireless support would represent the 52.3% of the consumption. However, broadcastable data will allow energy savings because the transceivers will not need to be active all the time slots, and we will reduce the off-chip wired overheads as well.

Component	**Sub-element**	**Area** (mm^2)	**Area (%)**	**Power** (mW)	**Power (%)**
Compute array		1994.5	72.89	48281.6	97.6
	Chiplet	4.64	82.95	89.9	47.67
	Receiver	0.955	17.05	98.7	52.33
Memory		741.77	27.11	1373.3	2.4
	SRAM	740	99.76	1190	86.66
	Transmitter	1.77	0.24	183.3	13.34
Total		2736.3	100	49471.6	100

Table 6.1: Accelerator consumption breakdown for a total of 256 chiplets

Table 6.1 shows a breakdown of the accelerator area and power consumption when using Eyeriss chiplets. Each chiplet contains 64 PEs, a shared memory and a small NoC, and it is augmented with a receiver. Then, the compute array consists of 256 of these chiplets.
As it is noticeable, these overheads are not negligible and may be an important drawback as well. However, we chose the 64 bytes bandwidth, which is quite a radical choice.

6.2 Budget

The objective of this thesis is not a physical implementation of the prototype, and therefore no purchasings have been done. It did not require the assembly of any hardware components, as the analysis was carried out using simulation tools. All the software that has been used for the development of this project, such as MAESTRO, is open source and therefore no license has been needed.
However, it required personnel and work hours. The working time has been eight hours per day, five days per week during five months. Thus, the total number of hours put is 800. By evaluating this hours of work at a junior engineer salary, given the approximation of $42,000/year in United States of America [108], and assuming a full-time job of 40 hours a week x 52 business weeks a year, a pay of $20.19/hour can be computed. Then, the human cost of this project can be approximated as $16,152.

[2]Assuming an average of 1cm links and a BER equal to 10^{-5}.

Should we develop this accelerator as a real product, only the prototype itself would cost around \$0.5M-\$1M as a rough estimate. First of all, we would need the design tool to explore the design space, which could be MAESTRO. Then, we would also need the RTL design software, as well as the libraries for synthesis and the Process Design Kit (PDK) files. Next, we would require the chip and wireless components, which would also incur into a package and assembly cost to integrate everything. To do so, at least five employees working a whole year would be needed, as an estimation, for the design and simulation of the dataflows, the chips, the wireless interconnects and the integration.
As it can be seen, the cost would be huge, but the impact of this technology would be proportional, since the improvement of the ML computation is required to bring the AI to our daily life.

7. Conclusions and future development

7.1 Conclusions

This project analyzed the techniques that are being used and will be used in the future of Artificial Intelligence inference in the edge by means of hardware acceleration. A study of the different dataflows in these accelerators has been carried out, and augmenting them with wireless networks at chip scale has been proposed.

The thesis is composed of seven chapters. The first one introduces the work environment, the planning and the initial ideas behind this project. Chapter two gives relevant background in artificial neural networks and how DNN accelerators handle them smoothly. The Wireless Networks-on-Chip are introduced and its context is explained. Chapter three motivates the work by describing dataflows, a concept of utmost importance in DNN accelerators, and the opportunities WNoC can have to enhance current approaches. Next, chapter four explains the procedures such as the software and the models used. Chapter five then proceeds to evaluate the design space, both on-chip and off-chip. Finally, a specific design point is proposed in Chapter six, and Chapter seven concludes.

The initial idea was to explore the possibilities of these two technologies - DNN accelerators and WNoCs - to work together. The required capabilities for the NoCs in NN accelerators have been first identified, by iterating different dataflows and observing their bandwidth requirements. After a first on-chip approach, I could not find an efficient way to embody wireless transceivers on-chip that would be more profitable than current NoCs. However, when the off-chip interconnects were analyzed, I found WNoCs to be potentially usable to enhance communication links at chiplet level, due to dataflow flexibility and increased bandwidth.
Finally, a precise scaled DNN accelerator with wireless interconnects has been proposed, simulated and evaluated. Results show a potential savings of 13X in runtime but at an important overhead cost.

7.2 Future work

Throughout this thesis, some points have been made about potential ideas to be developed. These are summarized here as well as new ones.

The apparent fit between DNN accelerators and WNoCs goes beyond the analysis done in this thesis. For instance, as mentioned in Chapter 2, layer-by-layer precision can leverage the neural network computational expense and wireless networks-on-chip can facilitate it due to its serializable nature. Similarly, the concept of *sparsity* is of great importance in neural networks and some connections may be found with WNoCs. I believe these two concepts applied to WNoC technology in accelerators may be useful to enhance them.

Another point that could be worth researching, from my point of view, is the hybrid wireless-wired networks implementation in DNN accelerators. As explained in Chapter 5, I find it highly appropriate and it may be a wise intermediate step before relying exclusively on a WNoC implementation in accelerators.

Finally, I could not find a generic DNN accelerator simulation tool that relied on cycle-accurate emulation for different NoCs. This is probably because the data movements in an accelerator are highly deterministic, but from a WNoC perspective it would have been nice to have, at least in my case, and it may be useful for the community as well.

Bibliography

[1] T. Krishna, "Synergy lab." [Online] https://synergy.ece.gatech.edu/. Accessed 2019-03-06.

[2] M. Evans, "Artificial Intelligence Is Expected To Be The Most Impactful Technology On Commerce," *Forbes*, 2019. [Online] https://www.forbes.com/sites/michelleevans1/2019/01/16/artificial-intelligence-is-expected-to-be-the-most-impactful-technology-on-commerce/. Accessed 2019-05-20.

[3] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *International Conference on Learning Representations*, Sept. 2014. arxivId:1409.1556.

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI:10.1109/5.726791.

[5] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, IEEE, May 2013. DOI:10.1109/ICASSP.2013.6638947.

[6] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 881–892, July 2002. DOI:10.1109/TPAMI.2002.1017616.

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *Conference and Workshop on Neural Information Processing Systems*, June 2014. arxivId:1406.2661.

[8] D. Ardila, A. P. Kiraly, S. Bharadwaj, B. Choi, J. J. Reicher, L. Peng, D. Tse, M. Etemadi, W. Ye, G. Corrado, D. P. Naidich, and S. Shetty, "End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography," *Nature Medicine*, p. 1, May 2019. DOI:10.1038/s41591-019-0447-x.

[9] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," 2017. arXiv:1704.01212.

[10] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic Image Synthesis with Spatially-Adaptive Normalization," *Computer Vision and Pattern Recognition*, Mar. 2019. arxivId:1903.07291.

[11] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Dec. 2015. DOI:10.1109/CVPR.2016.308.

[12] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958. DOI:10.1037/h0042519.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *Computer Vision and Pattern Recognition*, Sept. 2014. arxivId:1409.4842.

[14] "Tractica: NVIDIA's Inference Push for AI," *Tractica*. [Online] https://www.tractica.com/artificial-intelligence/nvidias-inference-push-for-ai/. Accessed 2019-06-01.

[15] "Semiconductor Engineering: AI Chip Architectures Race To The Edge," *Semiconductor Engineering*. [Online] https://semiengineering.com/ai-chip-architectures-race-to-the-edge/. Accessed 2019-04-05.

[16] M. Dixon, D. Klabjan, and J. H. Bang, "Classification-based Financial Markets Prediction using Deep Neural Networks," *Algorithmic Finance*, Mar. 2016. arxivId:1603.08604.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. DOI:10.1109/CVPR.2016.90.

[18] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits*, vol. 52, pp. 127–138, Jan. 2017. DOI:10.1109/JSSC.2016.2616357.

[19] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning," in *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Mar. 2014. DOI:10.1145/2541940.2541967.

[20] H. Kwon, A. Samajdar, and T. Krishna, "MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS*, vol. 53, (New York, New York, USA), pp. 461–475, ACM Press, Mar. 2018. DOI:10.1145/3173162.3173176.

[21] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016. ISBN:0262035618.

[22] A. Lazorenko, "TensorFlow performance test: CPU VS GPU," *Medium*, 2017. [Online] https://medium.com/@andriylazorenko/tensorflow-performance-test-cpu-vs-gpu-79fcd39170c. Accessed 2019-04-08.

[23] "Google TPU." [Online] https://cloud.google.com/tpu/docs/tpus. Accessed 2019-06-01.

[24] "Nervana Neural Network Processor. Intel AI." [Online] https://www.intel.ai/nervana-nnp/. Accessed 2019-05-25.

[25] T. Gale, E. Elsen, and S. Hooker, "The State of Sparsity in Deep Neural Networks," *arXiv*, Feb. 2019. arxivId:1902.09574.

[26] S. Alford, R. Robinett, L. Milechin, and J. Kepner, "Pruned and Structurally Sparse Neural Networks," *arXiv*, Sept. 2018. arxivId:1810.00299.

[27] S. Wang and T. Jin, "Wireless network-on-chip: a survey," *The Journal of Engineering*, vol. 2014, pp. 98–104, Mar. 2014. DOI:10.1049/joe.2013.0209.

[28] N. E. Jerger, T. Krishna, and L.-S. Peh, "On-Chip Networks, Second Edition," *Synthesis Lectures on Computer Architecture*, vol. 12, pp. 1–210, June 2017. DOI:10.2200/S00772ED1V01Y201704CAC040.

[29] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)*, pp. 684–689, June 2001. DOI:10.1109/DAC.2001.156225.

[30] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "SCNN: An Accelerator for Compressed-sparse Convolutional Neural Networks," *ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, June 2017. DOI:10.1145/3079856.3080254.

[31] V. Fernando, A. Franques, S. Abadal, S. Misailovic, and J. Torrellas, "Replica: A wireless manycore for communication-intensive and approximate data," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages*

and Operating Systems, ASPLOS '19, (New York, NY, USA), pp. 849–863, ACM, 2019. DOI:10.1145/3297858.3304033.

[32] C. Xiao and W. Liu, "Through Global Sharing to Improve Network Efficiency for Radio-Frequency Interconnect Based Network-on-Chip," *IEEE Access*, vol. 4, pp. 6503–6514, Oct. 2016. DOI:10.1109/ACCESS.2016.2611141.

[33] S. Deb and H. Mondal, "Wireless network-on-chip: a new era in multi-core chip design," in *2014 25nd IEEE International Symposium on Rapid System Prototyping*, pp. 59–64, IEEE, Oct. 2014. DOI:10.1109/RSP.2014.6966893.

[34] H. Yordanov, V. Poulkov, and P. Russer, "On-chip monolithic integrated antennas using cmos ground supply planes," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 6, pp. 1268–1275, Aug. 2016. DOI:10.1109/TCPMT.2016.2587663.

[35] J. Wu, A. K. Kodi, S. Kaya, A. Louri, and H. Xin, "Monopoles loaded with 3-d-printed dielectrics for future wireless intrachip communications," *IEEE Transactions on Antennas and Propagation*, vol. 65, pp. 6838–6846, Dec. 2017. DOI:10.1109/TAP.2017.2758400.

[36] T. S. Rappaport, *Millimeter wave wireless communications*. Prentice Hall, 2014. ISBN:9780132173636.

[37] A. Ganguly, K. Chang, S. Deb, P. P. Pande, B. Belzer, and C. Teuscher, "Scalable hybrid wireless network-on-chip architectures for multicore systems," *IEEE Transactions on Computers*, vol. 60, pp. 1485–1502, Oct. 2011. DOI:10.1109/TC.2010.176.

[38] S. Abadal, E. Alarcón, A. Cabellos-Aparicio, M. Lemme, and M. Nemirovsky, "Graphene-enabled wireless communication for massive multicore architectures," *IEEE Communications Magazine*, vol. 51, pp. 137–143, Nov. 2013. DOI:10.1109/MCOM.2013.6658665.

[39] S. Deb, A. Ganguly, K. Chang, P. Pande, B. Beizer, and D. Heo, "Enhancing performance of network-on-chip architectures with millimeter-wave wireless interconnects," in *ASAP 2010 - 21st IEEE International Conference on Application-specific Systems, Architectures and Processors*, pp. 73–80, IEEE, July 2010. DOI:10.1109/ASAP.2010.5540799.

[40] X. Timoneda, S. Abadal, A. Franques, D. Manessis, J. Zhou, J. Torrellas, E. Alarcón, and A. Cabellos-Aparicio, "Engineer the Channel and Adapt to it: Enabling Wireless Intra-Chip Communication," *arXiv*, Dec. 2018. arxivId:1901.04291.

[41] D. W. Matolak, S. Kaya, and A. Kodi, "Channel modeling for wireless networks-on-chips," *IEEE Communications Magazine*, vol. 51, pp. 180–186, June 2013. DOI:10.1109/MCOM.2013.6525613.

[42] X. Timoneda, S. Abadal, A. Cabellos-Aparicio, D. Manessis, J. Zhou, A. Franques, J. Torrellas, and E. Alarcón, "Millimeter-wave propagation within a computer chip package," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2018. DOI:10.1109/ISCAS.2018.8351875.

[43] S. Deb, K. Chang, X. Yu, S. P. Sah, M. Cosic, A. Ganguly, P. P. Pande, B. Belzer, and D. Heo, "Design of an energy-efficient cmos-compatible noc architecture with millimeter-wave wireless interconnects," *IEEE Transactions on Computers*, vol. 62, pp. 2382–2396, Dec. 2013. DOI:10.1109/TC.2012.224.

[44] S. Abadal, M. Iannazzo, M. Nemirovsky, A. Cabellos-Aparicio, H. Lee, and E. Alarcon, "On the Area and Energy Scalability of Wireless Network-on-Chip: A Model-Based Benchmarked Design Space Exploration," *IEEE/ACM Transactions on Networking*, vol. 23, pp. 1501–1513, Oct. 2015. DOI:10.1109/TNET.2014.2332271.

[45] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos, "Stripes: Bit-serial deep neural network computing," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–12, IEEE, Oct. 2016. DOI:10.1109/MICRO.2016.7783722.

[46] Xinmin Yu, J. Baylon, P. Wettin, Deukhyoun Heo, P. P. Pande, and S. Mirabbasi, "Architecture and Design of Multichannel Millimeter-Wave Wireless NoC," *IEEE Design & Test*, vol. 31, pp. 19–28, Dec. 2014. DOI:10.1109/MDAT.2014.2322995.

[47] S. Abadal, A. Marruedo, A. Franques, H. Taghvaee, A. Cabellos-Aparicio, J. Zhou, J. Torrellas, and E. Alarcon, "Opportunistic Beamforming in Wireless Network-on-Chip," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, May 2019. DOI:10.1109/ISCAS.2019.8702417.

[48] M. S. Shamim, N. Mansoor, A. Samaiyar, A. Ganguly, S. Deb, and S. Sunndar Ram, "Energy-efficient Wireless Network-on-chip Architecture with Log-periodic On-chip Antennas," pp. 85–86, May 2014. DOI:10.1145/2591513.2591566.

[49] V. Vijayakumaran, M. P. Yuvaraj, N. Mansoor, N. Nerurkar, A. Ganguly, and A. Kwasinski, "CDMA Enabled Wireless Network-on-Chip," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 10, pp. 1–20, June 2014. DOI:10.1145/2536778.

[50] C. Killebrew, *L2 Cache to Off-chip Memory Networks for Chip Multiprocessor*. PhD thesis, EECS Department, University of California, Berkeley, May 2008.

[51] S. Deb, A. Ganguly, P. P. Pande, B. Belzer, and D. Heo, "Wireless noc as interconnection backbone for multicore chips: Promises and challenges," *IEEE Journal on Emerging and Selected Top-*

ics in Circuits and Systems, vol. 2, pp. 228–239, June 2012. DOI:10.1109/JETCAS.2012.2193835.

[52] H. Kwon, P. Chatarasi, M. Pellauer, V. Sarkar, and T. Krishna, "A Data-Centric Approach for Modeling and Estimating Efficiency of Dataflows for Accelerator Design," May 2018. arxivId:1805.02566.

[53] M. Alwani, H. Chen, M. Ferdman, and P. Milder, "Fused-layer CNN accelerators," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–12, IEEE, Oct. 2016. DOI:10.1109/MICRO.2016.7783725.

[54] A. Parashar, P. Raina, Y. S. Shao, Y. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A systematic approach to dnn accelerator evaluation," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 304–315, Mar. 2019. DOI:10.1109/ISPASS.2019.00042.

[55] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient Inference Engine on Compressed Deep Neural Network," *International Symposium on Computer Architecture*, Feb. 2016. arxivId:1602.01528.

[56] D. DiTomaso, A. Kodi, D. Matolak, S. Kaya, S. Laha, and W. Rayess, "A-winoc: Adaptive wireless network-on-chip architecture for chip multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, pp. 3289–3302, Dec. 2015. DOI:10.1109/TPDS.2014.2383384.

[57] H. K. Mondal and S. Deb, "Energy efficient on-chip wireless interconnects with sleepy transceivers," in *2013 8th IEEE Design and Test Symposium*, pp. 1–6, Dec. 2013. DOI:10.1109/IDT.2013.6727078.

[58] A. Y. Wang and C. G. Sodini, "On the energy efficiency of wireless transceivers," in *IEEE International Conference on Communications*, June 2006. DOI:10.1109/ICC.2006.255661.

[59] P. Rodríguez-Vázquez, J. Grzyb, N. Sarmah, B. Heinemann, and U. R. Pfeiffer, "A 65 gbps qpsk one meter wireless link operating at a 225–255 ghz tunable carrier in a sige hbt technology," in *2018 IEEE Radio and Wireless Symposium (RWS)*, pp. 146–149, Jan. 2018. DOI:10.1109/RWS.2018.8304970.

[60] J. Pang, S. Maki, S. Kawai, N. Nagashima, Y. Seo, M. Dome, H. Kato, M. Katsuragi, K. Kimura, S. Kondo, Y. Terashima, H. Liu, T. Siriburanon, A. Tharayil Narayanan, N. Fajri, T. Kaneko, T. Yoshioka, B. Liu, Y. Wang, and K. Okada, "A 50.1-gb/s 60-ghz cmos transceiver for ieee 802.11ay with calibration of lo feedthrough and i/q imbalance," *IEEE Journal of Solid-State Circuits*, vol. PP, pp. 1–16, Jan. 2019. DOI:10.1109/JSSC.2018.2886338.

[61] X. Yu, S. P. Sah, H. Rashtian, S. Mirabbasi, P. P. Pande, and D. Heo, "A 1.2-pj/bit 16-gb/s 60-ghz ook transmitter in 65-nm cmos for wireless network-on-chip," *IEEE Transactions on Microwave Theory and Techniques*, vol. 62, pp. 2357–2369, Oct. 2014. DOI:10.1109/TMTT.2014.2347919.

[62] X. Yu, H. Rashtian, S. Mirabbasi, P. P. Pande, and D. Heo, "An 18.7-gb/s 60-ghz ook demodulator in 65-nm cmos for wireless network-on-chip," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, pp. 799–806, Mar. 2015. DOI:10.1109/TCSI.2014.2386751.

[63] Y. Kim, Y. Du, A. Tang, Y. Zhao, B. Lee, H. Chen, C. Jou, J. Cong, T. Itoh, and M. F. Chang, "A 125ghz transceiver in 65nm cmos assembled with fr4 pcb antenna for contactless wave-connectors," in *2017 IEEE MTT-S International Microwave Symposium (IMS)*, pp. 1535–1538, June 2017. DOI:10.1109/MWSYM.2017.8058920.

[64] H. J. Lee, J. G. Lee, C. J. Lee, T. H. Jang, H. J. Kim, and C. S. Park, "High-speed and low-power ook cmos transmitter and receiver for wireless chip-to-chip communication," in *2015 IEEE MTT-S International Microwave Workshop Series on Advanced Materials and Processes for RF and THz Applications (IMWS-AMP)*, pp. 1–3, July 2015. DOI:10.1109/IMWS-AMP.2015.7324964.

[65] N. Dolatsha, B. Grave, M. Sawaby, C. Chen, A. Babveyh, S. Kananian, A. Bisognin, C. Luxey, F. Gianesello, J. Costa, C. Fernandes, and A. Arbabian, "17.8 a compact 130ghz fully packaged point-to-point wireless system with 3d-printed 26dbi lens antenna achieving 12.5gb/s at 1.55pj/b/m," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 306–307, Feb. 2017. DOI:10.1109/ISSCC.2017.7870383.

[66] K. Okada, R. Minami, Y. Tsukui, S. Kawai, Y. Seo, S. Sato, S. Kondo, T. Ueno, Y. Takeuchi, T. Yamaguchi, A. Musa, R. Wu, M. Miyahara, and A. Matsuzawa, "20.3 a 64-qam 60ghz cmos transceiver with 4-channel bonding," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 346–347, Feb. 2014. DOI:10.1109/ISSCC.2014.6757463.

[67] C. Thakkar, S. Shopov, A. Chakrabarti, S. Yamada, D. Choudhury, J. Jaussi, and B. Casper, "9.6 a 42.2gb/s 4.3pj/b 60ghz digital transmitter with 12b/symbol polarization mimo," in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, pp. 172–174, Feb. 2019. DOI:10.1109/ISSCC.2019.8662501.

[68] N. Weissman, S. Jameson, and E. Socher, "An f-band dual band 12+12gbps packaged cmos bi-directional transceiver," in *2016 IEEE MTT-S International Microwave Symposium (IMS)*, pp. 1–4, May 2016. DOI:10.1109/MWSYM.2016.7540217.

[69] K. Zhan, A. Agrawal, M. Johnson, A. Ramachandran, T. Anand, and A. Natarajan, "An integrated 7-gb/s 60-ghz communication link over single conductor wire using sommerfeld wave

propagation in 65-nm cmos," in *2017 IEEE MTT-S International Microwave Symposium (IMS)*, pp. 797–800, June 2017. DOI:10.1109/MWSYM.2017.8058698.

[70] Y. Tanaka, Y. Hino, Y. Okada, T. Takeda, S. Ohashi, H. Yamagishi, K. Kawasaki, and A. Hajimiri, "A versatile multi-modality serial link," in *2012 IEEE International Solid-State Circuits Conference*, pp. 332–334, Feb. 2012. DOI:10.1109/ISSCC.2012.6177034.

[71] S. Geng, D. Liu, Y. Li, H. Zhuo, W. Rhee, and Z. Wang, "A 13.3 mw 500 mb/s ir-uwb transceiver with link margin enhancement technique for meter-range communications," *IEEE Journal of Solid-State Circuits*, vol. 50, pp. 669–678, Mar. 2015. DOI:10.1109/JSSC.2015.2393815.

[72] D. Fritsche, P. Stärke, C. Carta, and F. Ellinger, "A low-power sige bicmos 190-ghz transceiver chipset with demonstrated data rates up to 50 gbit/s using on-chip antennas," *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, pp. 3312–3323, Sept. 2017. DOI:10.1109/TMTT.2017.2677908.

[73] K. Kawasaki, Y. Akiyama, K. Komori, M. Uno, H. Takeuchi, T. Itagaki, Y. Hino, Y. Kawasaki, K. Ito, and A. Hajimiri, "A millimeter-wave intra-connect solution," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 2655–2666, Dec. 2010. DOI:10.1109/JSSC.2010.2077130.

[74] P. Rodríguez- Vázquez, J. Grzyb, N. Sarmah, B. Heinemann, and U. R. Pfeiffer, "Towards 100 gbps: A fully electronic 90 gbps one meter wireless link at 230 ghz," in *2018 48th European Microwave Conference (EuMC)*, pp. 1389–1392, Sept. 2018. DOI:10.23919/EuMC.2018.8541410.

[75] X. Yu, S. P. Sah, S. Deb, P. P. Pande, B. Belzer, and D. Heo, "A wideband body-enabled millimeter-wave transceiver for wireless network-on-chip," in *2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1–4, Aug. 2011. DOI:10.1109/MWSCAS.2011.6026282.

[76] S. Daneshgar, K. Dasgupta, C. Thakkar, A. Chakrabarti, S. Yamada, D. Choudhury, J. Jaussi, and B. Casper, "A 27.8gb/s 11.5pj/b 60ghz transceiver in 28nm cmos with polarization mimo," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pp. 166–168, Feb. 2018. DOI:10.1109/ISSCC.2018.8310236.

[77] Y. Kim, B. Hu, Y. Du, A. Tang, H. Chen, C. Jou, J. Cong, T. Itoh, and M. F. Chang, "A 20gb/s 79.5mw 127ghz cmos transceiver with digitally pre-distorted pam-4 modulation for contactless communications," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pp. 278–280, Feb. 2018. DOI:10.1109/ISSCC.2018.8310292.

[78] M. Fujishima, M. Motoyoshi, K. Katayama, K. Takano, N. Ono, and R. Fujimoto, "98 mw 10 gbps wireless transceiver chipset with d-band coms circuits," *IEEE Journal of Solid-State*

Circuits, vol. 48, no. 10, pp. 2273–2284, 2013. DOI:10.1109/JSSC.2013.2261192.

[79] C. W. Byeon, C. H. Yoon, and C. S. Park, "A 67-mw 10.7-gb/s 60-ghz ook cmos transceiver for short-range wireless communications," *IEEE Transactions on Microwave Theory and Techniques*, vol. 61, pp. 3391–3401, Sept. 2013. DOi:10.1109/TMTT.2013.2274962.

[80] Y. Kim, S. Tam, T. Itoh, and M. F. Chang, "A 60-ghz cmos transceiver with on-chip antenna and periodic near field directors for multi-gb/s contactless connector," *IEEE Microwave and Wireless Components Letters*, vol. 27, pp. 404–406, Apr. 2017. DOI:10.1109/LMWC.2017.2678444.

[81] F. Zhu, W. Hong, W. Liang, J. Chen, X. Jiang, P. Yan, and K. Wu, "A low-power low-cost 45-ghz ook transceiver system in 90-nm cmos for multi-gb/s transmission," *IEEE Transactions on Microwave Theory and Techniques*, vol. 62, pp. 2105–2117, Sept. 2014. DOI:10.1109/TMTT.2014.2338276.

[82] K. Nakajima, A. Maruyama, M. Kohtani, T. Sugiura, E. Otobe, J. Lee, S. Cho, K. Kwak, J. Lee, T. Yoshimasu, and M. Fujishima, "23gbps 9.4pj/bit 80/100ghz band cmos transceiver with on-board antenna for short-range communication," in *2014 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pp. 173–176, Nov. 2014. DOI:10.1109/ASSCC.2014.7008888.

[83] K. K. Tokgoz, S. Maki, J. Pang, N. Nagashima, I. Abdo, S. Kawai, T. Fujimura, Y. Kawano, T. Suzuki, T. Iwai, K. Okada, and A. Matsuzawa, "A 120gb/s 16qam cmos millimeter-wave wireless transceiver," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pp. 168–170, Feb. 2018. DOI:10.1109/ISSCC.2018.8310237.

[84] S. Foulon, S. Pruvost, D. Pache, C. Loyez, and N. Rolland, "A 140 ghz multi-gigabits self-heterodyne transceiver for chip-to-chip communications," in *2014 44th European Microwave Conference*, pp. 901–904, Oct. 2014. DOI:10.1109/EuMC.2014.6986581.

[85] J. Lee, Y. Chen, and Y. Huang, "A low-power low-cost fully-integrated 60-ghz transceiver system with ook modulation and on-board antenna assembly," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 264–275, Feb. 2010. DOI:10.1109/JSSC.2009.2034806.

[86] "Cenaero: Pareto Front," *Cenaero*. [Online] http://www.cenaero.be/Page.asp?docid=27103. Accessed 2019-03-10.

[87] S. Upadhyay and S. Srivastava, "A 60-GHz on-chip monopole antenna using silicon technology," in *2013 IEEE Applied Electromagnetics Conference (AEMC)*, pp. 1–2, IEEE, Dec. 2013. DOI:10.1109/AEMC.2013.7045122.

[88] J. Gorisse, D. Morche, and J. Jantunen, "Wireless transceivers for gigabit-per-second communications," in *10th IEEE International NEWCAS Conference*, pp. 545–548, IEEE, June 2012. DOI:10.1109/NEWCAS.2012.6329077.

[89] I. Shomorony and A. S. Avestimehr, "Worst-case additive noise in wireless networks," *IEEE Transactions on Information Theory*, vol. 59, pp. 3833–3847, June 2013. DOI:10.1109/TIT.2013.2248875.

[90] A. Sharifi, E. Kultursay, M. Kandemir, and C. R. Das, "Addressing End-to-End Memory Access Latency in NoC-Based Multicores," in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 294–304, IEEE, Dec. 2012. DOI:10.1109/MICRO.2012.35.

[91] J. Engel and T. Kocak, "Off-chip communication architectures for high throughput network processors," *Computer Communications*, vol. 32, pp. 867–879, Mar. 2009. DOI:10.1016/J.COMCOM.2008.12.043.

[92] D. Sanchez, G. Michelogiannakis, and C. Kozyrakis, "An analysis of on-chip interconnection networks for large-scale chip multiprocessors," *ACM Transactions on Architecture and Code Optimization*, vol. 7, pp. 1–28, Apr. 2010. DOI:10.1145/1736065.1736069.

[93] S. Abadal, B. Sheinman, O. Katz, O. Markish, D. Elad, Y. Fournier, D. Roca, M. Hanzich, G. Houzeaux, M. Nemirovsky, E. Alarcón, and A. Cabellos-Aparicio, "Broadcast-enabled massive multicore architectures: A wireless rf approach," *IEEE Micro*, vol. 35, pp. 52–61, Sept. 2015. DOI:10.1109/MM.2015.123.

[94] J. Kim, K. Choi, and G. Loh, "Exploiting new interconnect technologies in on-chip communication," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, pp. 124–136, June 2012. DOI:10.1109/JETCAS.2012.2201031.

[95] D. Stow, I. Akgun, R. Barnes, P. Gu, and Y. Xie, "Cost analysis and cost-driven IP reuse methodology for SoC design based on 2.5D/3D integration," in *Proceedings of the 35th International Conference on Computer-Aided Design - ICCAD '16*, (New York, New York, USA), pp. 1–6, ACM Press, 2016. DOI:10.1145/2966986.2980095.

[96] J. Kim, M. Lee, H. M. Torun, K. Roy, M. Swaminathan, S. Mukhopadhyay, T. Krishna, S. K. Lim, G. Murali, H. Park, E. Qin, H. Kwon, V. Chaitanya, K. Chekuri, N. Dasari, and A. Singh, "Architecture, Chip, and Package Co-design Flow for 2.5D IC Design Enabling Heterogeneous IP Reuse," in *Proceedings of the 56th Annual Design Automation Conference 2019 on - DAC '19*, (New York, New York, USA), pp. 1–6, ACM Press, Jan. 2019. DOI:10.1145/3316781.3317775.

[97] M. S. Shamim, M. M. Ahmed, N. Mansoor, and A. Ganguly, "Energy-efficient wireless interconnection framework for multichip systems with in-package memory stacks," in *2017 30th IEEE International System-on-Chip Conference (SOCC)*, pp. 357–362, Sept. 2017. DOI:10.1109/SOCC.2017.8226077.

[98] M. A. I. Sikder, *Emerging Technologies in On-Chip and Off-Chip Interconnection Network*. PhD thesis, Ohio University, 2016.

[99] M. A. I. Sikder, A. Kodi, W. Rayess, D. DiTomaso, D. Matolak, and S. Kaya, "Exploring wireless technology for off-chip memory access," in *2016 IEEE 24th Annual Symposium on High-Performance Interconnects (HOTI)*, pp. 92–99, Aug. 2016. DOI:10.1109/HOTI.2016.026.

[100] N. E. Jerger, A. Kannan, Z. Li, and G. H. Loh, "Noc architectures for silicon interposer systems: Why pay for more wires when you can get them (from your interposer) for free?," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 458–470, Dec. 2014. DOI:10.1109/MICRO.2014.61.

[101] S. H. Gade, H. K. Mondal, and S. Deb, "High bandwidth off-chip memory access through hybrid switching and inter-chip wireless links," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 100–105, July 2018. DOI:10.1109/ISVLSI.2018.00028.

[102] M. Sinha, S. H. Gade, W. Singh, and S. Deb, "Data-flow Aware CNN Accelerator with Hybrid Wireless Interconnection," in *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 1–4, IEEE, July 2018. DOI:10.1109/ASAP.2018.8445126.

[103] S. Saxena, G. Shu, R. K. Nandwana, M. Talegaonkar, A. Elkholy, T. Anand, W. Choi, and P. K. Hanumolu, "A 2.8 mw/gb/s, 14 gb/s serial link transceiver," *IEEE Journal of Solid-State Circuits*, vol. 52, pp. 1399–1411, May 2017. DOI:10.1109/JSSC.2016.2645738.

[104] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "Shidiannao: Shifting vision processing closer to the sensor," in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, pp. 92–104, June 2015. DOI:10.1145/2749469.2750389.

[105] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks," in *Proceedings of the 45th Annual International Symposium on Computer Architecture*, ISCA '18, (Piscataway, NJ, USA), pp. 764–775, IEEE Press, June 2018. DOi:10.1109/ISCA.2018.00069.

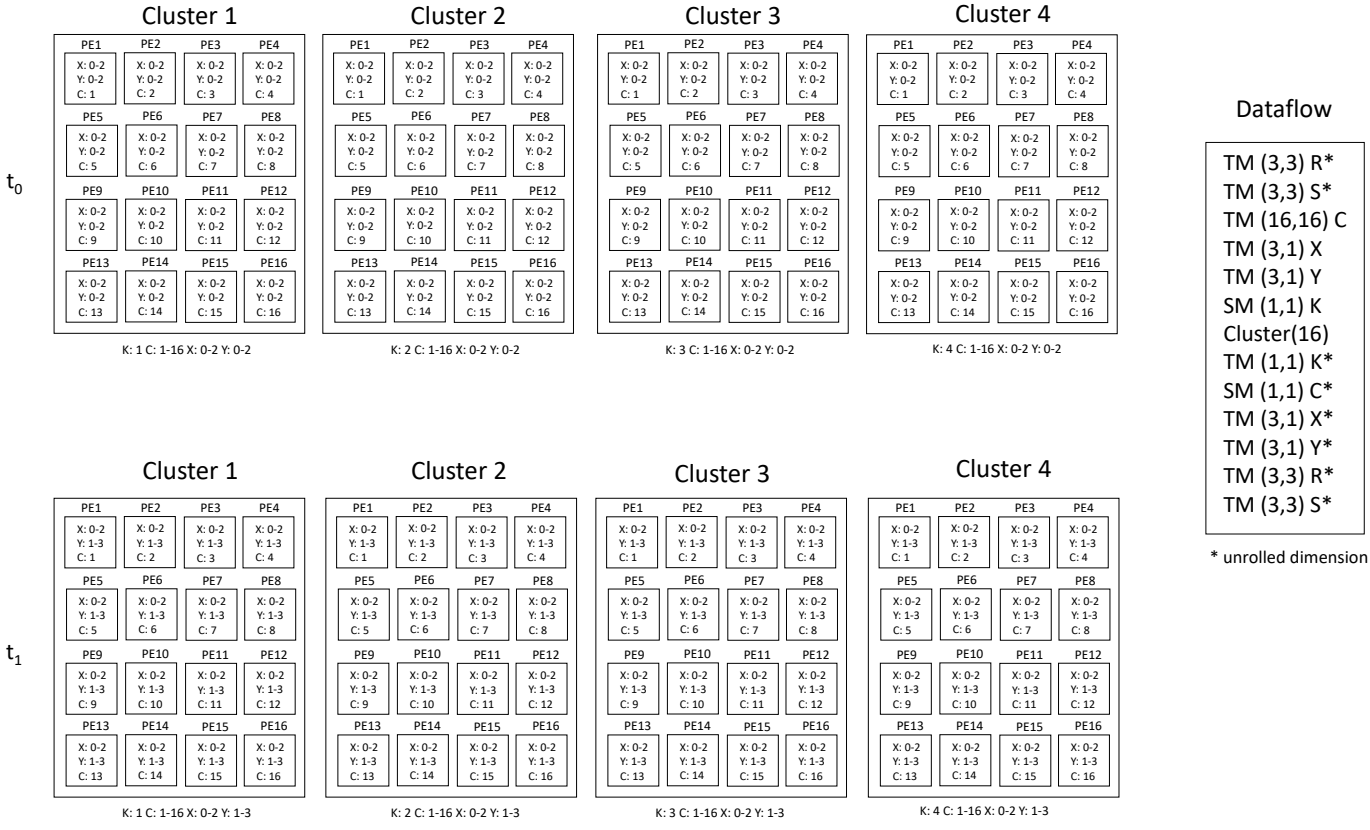
[106] C. Xin, Q. Chen, M. Tian, M. Ji, C. Zou, X. Wang, and B. Wang, "Cosy: An energy-efficient hardware architecture for deep convolutional neural networks based on systolic array," in *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 180–189, Dec. 2017. DOI:10.1109/ICPADS.2017.00034.

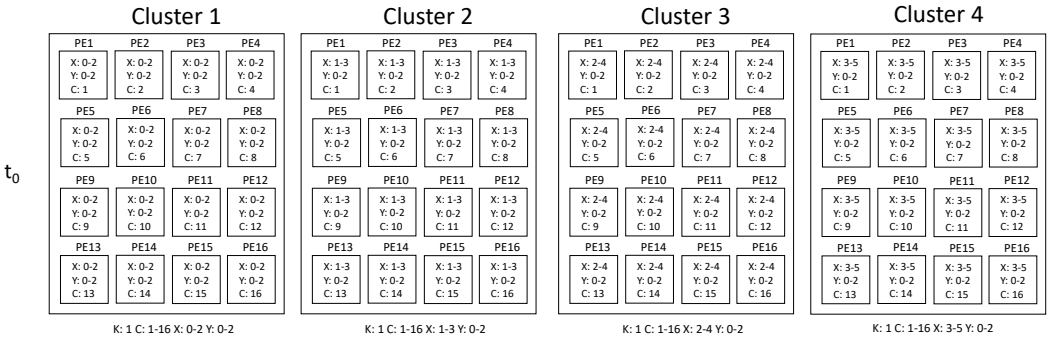
[107] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, "Tetris: Scalable and efficient neural network acceleration with 3d memory," *SIGARCH Comput. Archit. News*, vol. 45, pp. 751–764, Apr. 2017. DOI:10.1145/3093337.3037702.

[108] "PayScale: Junior Software Engineer Salary." [Online] https://www.payscale.com/research/US/Job=Junior_Software_Engineer/Salary. Accessed 2019-06-09.

Annex I

The following are exhaustively mapped dataflows. As an extension of the examples shown in the previous sections, these dataflows contain cluster levels. Both the MAESTRO dataflow description as well as the cluster and PEs mapping are represented. They show different behaviors due to different order and tiling sizes, which for instance change the latency hiding, the unrolled iterations or the broadcast opportunities.



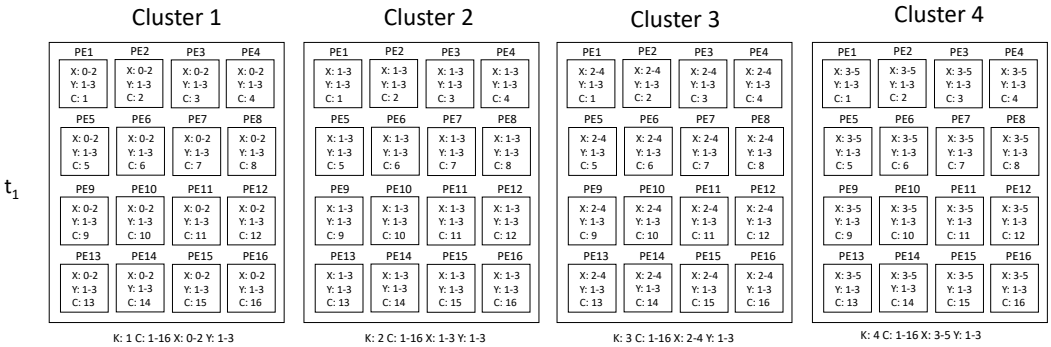


t_0

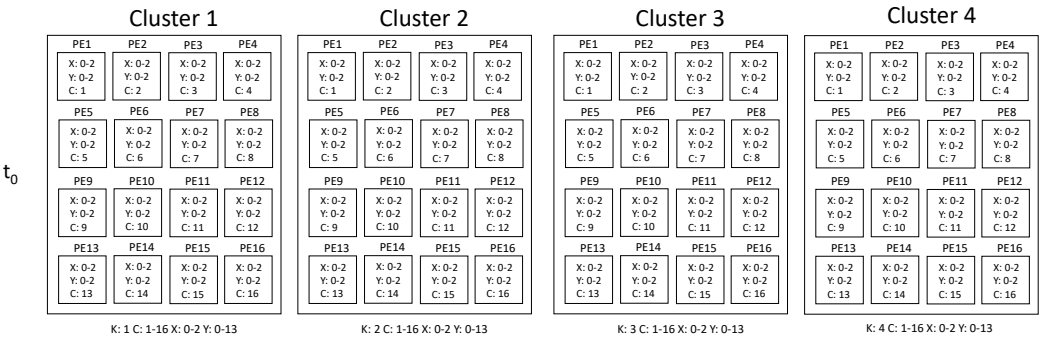
Dataflow

TM (3,3) R*
TM (3,3) S*
TM (1,1) K
TM (16,16) C
TM (3,1) Y
SM (3,1) X
Cluster(16)
TM (3,1) X*
TM (3,1) Y*
TM (1,1) K*
SM (1,1) C*
TM (3,3) R*
TM (3,3) S*

* unrolled dimension



t_1

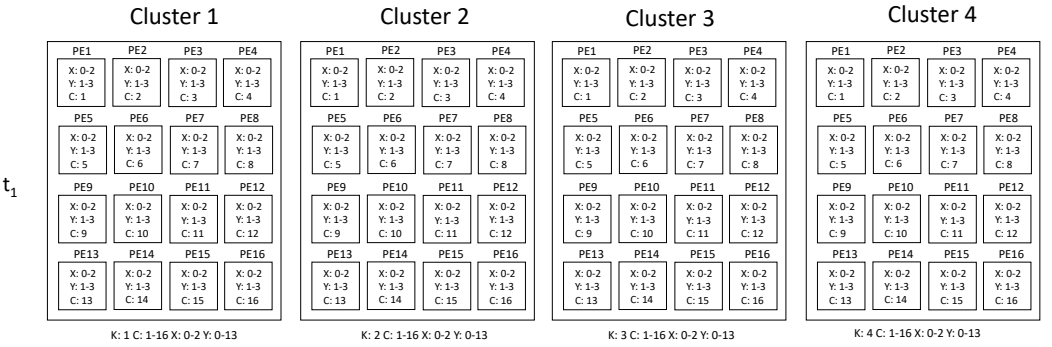


t_0

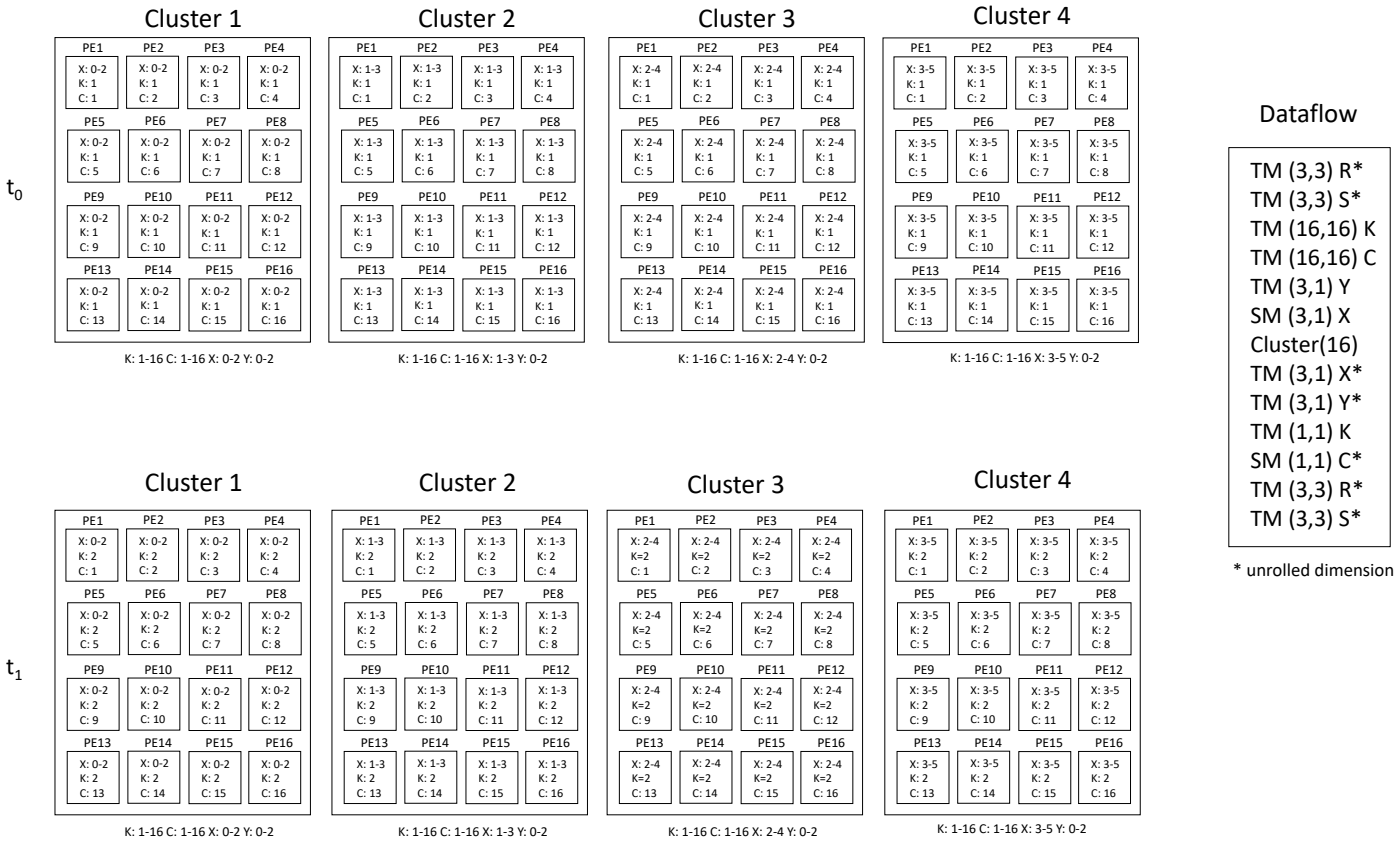
Dataflow

TM (3,3) R*
TM (3,3) S*
TM (16,16) C
TM (3,1) X
TM (14,12) Y
SM (1,1) K
Cluster(16)
TM (1,1) K*
SM (1,1) C*
TM (3,1) X*
TM (3,1) Y
TM (3,3) R*
TM (3,3) S*

* unrolled dimension



t_1



Note: If the most fine-grained tile to be temporally updated was the dimension C, we would not have any reuse opportunities since it is a coupled dimension with both weights and filters. Therefore, even though it is a valid dataflow, if used it would show poor performance.

Annex II

Revision history and approval record

Revision	**Date**	**Purpose**
0	12/05/2019	Document creation
1	10/06/2019	Document revision
2	15/06/2019	Document revision

DOCUMENT DISTRIBUTION LIST

Name	**e-mail**
Robert Guirado	roberto.guirado.linan@estudiant.upc.edu
Sergi Abadal	abadal@ac.upc.edu
Tushar Krishna	tushar@ece.gatech.edu

Written by:		Reviewed and approved by:	
Date	24/06/2019	Date	24/06/2019
Name	Robert Guirado	Name	Sergi Abadal
Position	Project Author	Position	Project Supervisor

Acronyms

AI Artificial Intelligence.

CNN Convolutional Neural Network.

DNN Deep Neural Network.

IS Input Stationary.

ML Machine Learning.

NLR No Local Reuse.

NN Neural Network.

OS Output Stationary.

PE Processing Element.

RS Row Stationary.

WNoC Wireless Network-on-Chip.

WS Weight Stationary.